UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA MECÂNICA E DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

Pedro Artur Ferreira Dos Santos Costa

SIMULAÇÃO DO COMPORTAMENTO DE UM ROBÔ MÓVEL NÃO COMERCIAL EM AMBIENTE DE ARMAZÉM 2D

GOIÂNIA

2019

sistema de bibliotecas ufg

PRPG
PRÓ-REITORIA DE PÓS-GRADUAÇÃO

UFG

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do material bibliográfico: [ ] Dissertação [ ] Tese

2. Identificação da Tese ou Dissertação:

Nome completo do autor: Pedro Artur Ferreira Dos Santos Costa

Título do trabalho: Simulação do comportamento de um robô móvel não comercial em ambiente de armazém 2d

3. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF da tese ou dissertação.

Pedro Artur Ferreira
Assinatura do(a) autor(a)[2]

Ciente e de acordo:

João Paulo da Silva Fonseca
Assinatura do(a) orientador(a)[2]

Data: 20 / 12 / 19

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. A extensão deste prazo suscita justificativa junto à coordenação do curso. Os dados do documento não serão disponibilizados durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

[2] A assinatura deve ser escaneada.

Pedro Artur Ferreira Dos Santos Costa

SIMULAÇÃO DO COMPORTAMENTO DE UM ROBÔ MÓVEL NÃO COMERCIAL EM AMBIENTE DE ARMAZÉM 2D

Monografia do projeto final apresentada como requisito parcial para obtenção do título de bacharel em Engenharia de Computação pela Universidade Federal de Goiás.

Orientador: Prof. Dr. João Paulo da Silva Fonseca

GOIÂNIA

2019

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Costa, Pedro Artur Ferreira dos Santos
Simulação do comportamento de um robô móvel não comercial em ambiente de armazém 2d [manuscrito] : / Pedro Artur Ferreira dos Santos Costa. - 2019.
xvi, 65 f.: il.

Orientador: Prof. Dr. João Paulo da Silva Fonseca.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia da Computação, Goiânia, 2019.
Bibliografia. Anexos. Apêndice.
Inclui lista de figuras.

1. Simulação. 2. V-REP. 3. Controle externo. 4. Robótica. 5. Armazenamento. I. Fonseca, João Paulo da Silva, orient. II. Título.

CDU 004

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE ESTÁGIO E PROJETO FINAL

ATA DE AVALIAÇÃO DE PROJETO FINAL

CURSO

( ) Eng. Elétrica ( ) Eng. Mecânica ( ) Eng. de Computação
( ) Projeto Final 1 (X) Projeto Final II

AVALIAÇÃO DE PROJETO FINAL

Título do projeto: Simulação do comportamento de robô móvel não comercial em ambiente de armazém 2D

BANCA AVALIADORA

Membro 1: João Paulo da Silva Fonseca
Membro 2: Carlos Galvão Pinheiro Júnior
Membro 3: Marco Antonio Assfalk de Oliveira

ESTUDANTES

Matrícula	Nome
201307409	Pedro Artur Ferreira dos Santos Costa

NOTAS

Matrícula	Membro 1				Membro 2				Membro 3				Média
	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	
201307409	9,0	5,0	8,0	6,8	9,0	4,0	8,0	6,3	9,0	4,0	8,0	6,3	6,5

NPT – Nota plano de trabalho; NTE – Nota do trabalho escrito; NAA – Nota de apresentação e arguição

Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA

Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

Goiânia, 13 de dezembro de 2019.

João Paulo da Silva Fonseca
Membro 1

Carlos Galvão Pinheiro Jr
Membro 2

Marco Antonio Assfalk de Oliveira
Membro 3

“Que não se admita no coração outro desejo ou propósito,
cuja o objeto supremo não seja ELE.”

AGRADECIMENTOS

Agradeço primeiramente a Deus, por sua força e auxílio. Agradeço aos meus pais Raimundo Ferreira dos Santos e Cleidia Francisca da Costa, pela paciência com as madrugadas andando pela casa e pelo suporte no decorrer do curso. Juntamente agradeço a todos companheiros dessa jornada, amigos e colegas feitos durante o curso, que foram essenciais para conclusão desse curso.

Agradeço ao pastor e amigo Luiz Fernando que me motivou a permanecer no curso que hoje se finaliza e também agradeço ao meu orientador Prof. Dr. João Paulo da Silva Fonseca, pela paciência, orientação, postura e auxílio durante todo trabalho.

RESUMO

Um dos grandes aprimoramentos da informática foi possibilitar a automação na busca por informação. Por meio da robótica, algumas empresas buscam trazer essa automação para o fluxo de produtos e materiais, buscando reduzir os custos com atividades de logística. Tendo como exemplo a Amazon©, que se tornou em 2017 a empresa mais valiosa do mundo se destacando justamente na área de automação logística, e buscando contribuir com uma base para simulações e trabalhos acadêmicos associados, este trabalho busca apresentar tarefas de navegação de robôs móveis *open-source*, apresentados recentemente na literatura, em um ambiente de armazém no software de simulação robótica V-REP. Busca-se que esse este ambiente seja autônomo e modular, no qual o robô simulado utiliza detectores de linha para se deslocar ao longo do percurso e etiquetas NFC(*Near Field Communication*) para compreender sua posição. A lógica de controle do robô se dá por meio de uma API remota desenvolvida em Python, que possibilita uma interface mais intuitiva para o usuário dos comandos. Os resultados alcançados refletem um modelo de simulação com controle de armazém autônomo e modular, no qual a interferência do usuário é unicamente a solicitação pelo número do armário.

Palavras chaves:
Simulação, V-REP, Controle externo, Robótica, Armazenamento

ABSTRACT

One of the great improvements of computing was to enable automation in the search for information. Through robotics, some companies seek to bring this automation to the flow of products and materials, seeking to reduce costs with logistics activities. Taking Amazon ©, which became the most valuable company in the world in 2017, standing out precisely in the area of logistics automation, and seeking to contribute with a basis for simulations and associated academic work, this paper aims to present mobile robot navigation tasks. open-source, recently presented in the literature, in a warehouse environment using the V-REP robotic simulation software. This environment is intended to be autonomous and modular, in which the simulated robot uses line detectors to travel along the path and Near Field Communication (NFC) tags to understand its position. The control logic of the robot is provided through a remote API developed in Python, which allows a more intuitive user interface of commands. The results achieved reflect a stand-alone, modular warehouse control simulation model in which user interference is only requested by cabinet number

Keywords:
Simulation, V-REP, External Control, Robotics, Warehousing

Lista de figuras

Sumário Página

1 Introdução

A Brand Finance divulgou relatório que lista as marcas mais valiosas e as marcas mais fortes do mundo. Em 2019, a Amazon manteve a posição número um, sendo eleita a mais valiosa. Com um crescimento de 24,6% em relação a 2018, a companhia de Jeff Bezos, homem mais rico do mundo desde 2017 quando ultrapassou o Bill gates, tendo no ano de 2019 um valor de marca de US$187,9 bilhões. Entre os fatores que dão tal destaque a esta empresa está a sua rápida realocação de recursos ou de objetos, “com dezenas de milhares de pedidos sendo feitos online simultaneamente você começa a precisar de algo além do que um humano pode fazer”, disse Tye Brady, chefe de tecnologia na Amazon, em entrevista ao MIT Technology Review. Disse também que os mais de 100 mil robôs autônomos em atividade nos galpões da empresa estão, na verdade, “expandindo a eficiência humana e não roubando empregos”. (Winick, 2018).

Outro fator que pode justificar um progresso econômico contínuo, como pode ser visto na figura 1, é a capacidade de inovação e foco na melhoria de serviços através da automatização. Como citado pela Pollux, empresa nacional de automação e robótica, ”Obviamente, a maior parte de suas estratégias estão guardadas a sete chaves, mas talvez a principal razão para um crescimento tão sólido esteja bem visível aos olhos de quem observa: a Amazon foi uma das pioneiras a estudar e implantar novas tecnologias no setor de distribuição e logística”. A automação nos armazéns ocorre principalmente através dos Robôs Kiva, originários de uma startup de robótica comprada pela Amazon, cada unidade do Kiva é capaz de movimentar mais de 1300 quilogramas (Bloglogística, 2015). Eles movimentam produtos aos empregados da Amazon para repor estoque ou retirada de algum item quando uma compra é realizada.

A distribuição das encomendas por serviço de robôs autônomos economiza em média 40 minutos do serviço de um ser humano por robô, com projetos da construção de um armazém completamente autônomo (POLLUX, 2018).

Este trabalho tem como área técnica a robótica móvel e visa a simulação de um ambiente de simulação robótica na distribuição de um armazém. Desenvolve-se um programa via API remota para controlar o movimento dos robôs para minimização de interferência humana.

Figura 1: Crescimento da Amazon nos últimos sete anos

Fonte: ADVFN (2019)

1.1. Motivação

A automação logística utilizada na simulação já é aplicada em diversos galpões, como exemplo pode ser visto na figura 2 o armazém da Amazon na Califórnia com três mil robôs em operação. Estruturas robóticas recebem uma orientação externa sobre sua posição e o caminho que deve ser tomado e, além do controle externo, um conjunto de sensores do

próprio robô, que por projeto se sobrepõem aos comandos da rede, impedem colisões imediatas.

Junto com o crescimento da inteligência artificial, da automação e dos robôs nas indústrias, surgiu o termo apontado por especialistas, como Luciano Floridi em The Fourth Revolution (apud, Átila Amarinho (2019)), como a quarta Revolução Industrial, que promete transformar profundamente a indústria, não apenas pela ampliação no uso de robôs, mas também pela ligação desses dispositivos em rede, pelo uso de sensores e pelo tratamento de dados em grande escala. Consequentemente, essa inteligência artificial se estende à logística e à cadeia de suprimentos.

Figura 2: Fotografia de um armazém automático da Amazon localizado no estado da Califórnia, demonstrando robôs Kiva em operação

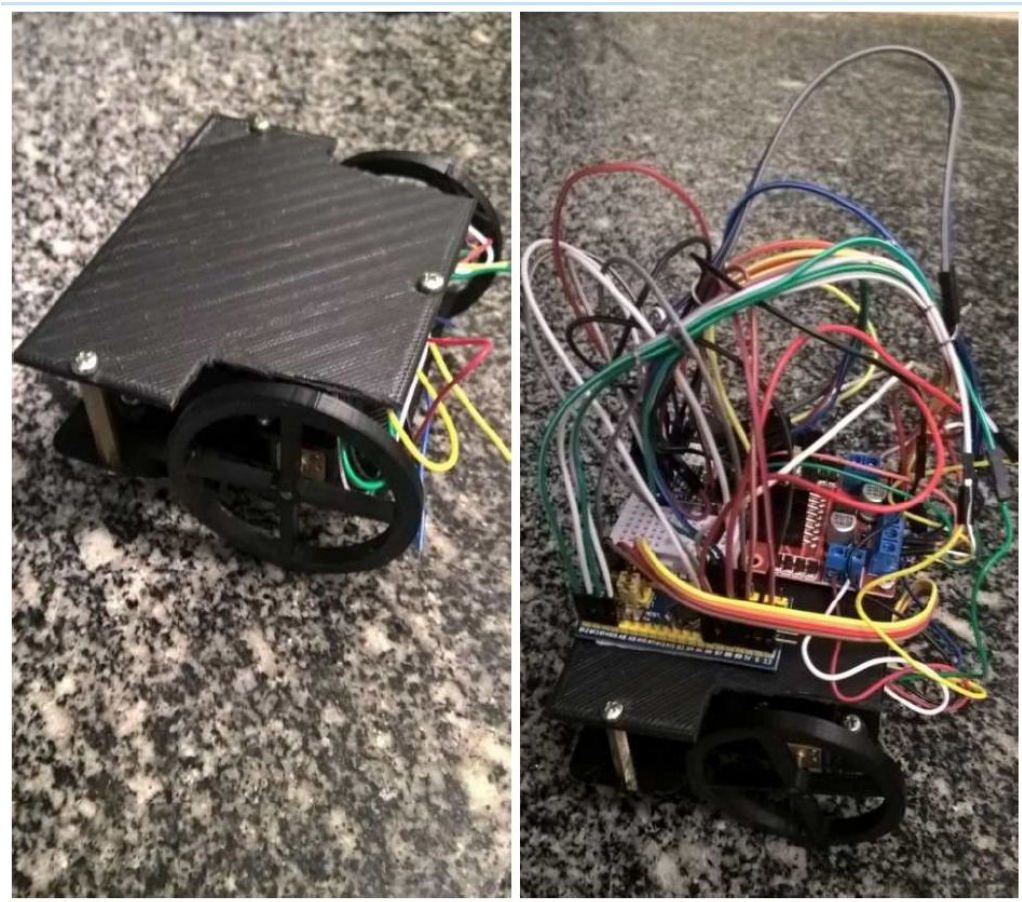
Fonte: Lobo (2016)

1.2. Justificativa

Este trabalho aborda a simulação de um ambiente de armazém e leva em consideração a utilização de robôs móveis *open-source*, vistos nas figuras 3 e 4, apresentados previamente por Ferreira et al. (2018) e por Fonseca (2018). Esta escolha se deu pelo fato destes modelos serem de fácil fabricação através de processos de fabricação com impressoras 3D, componentes que os robôs utilizam podem ser vistos na figura 5 . Observa-se a possibilidade

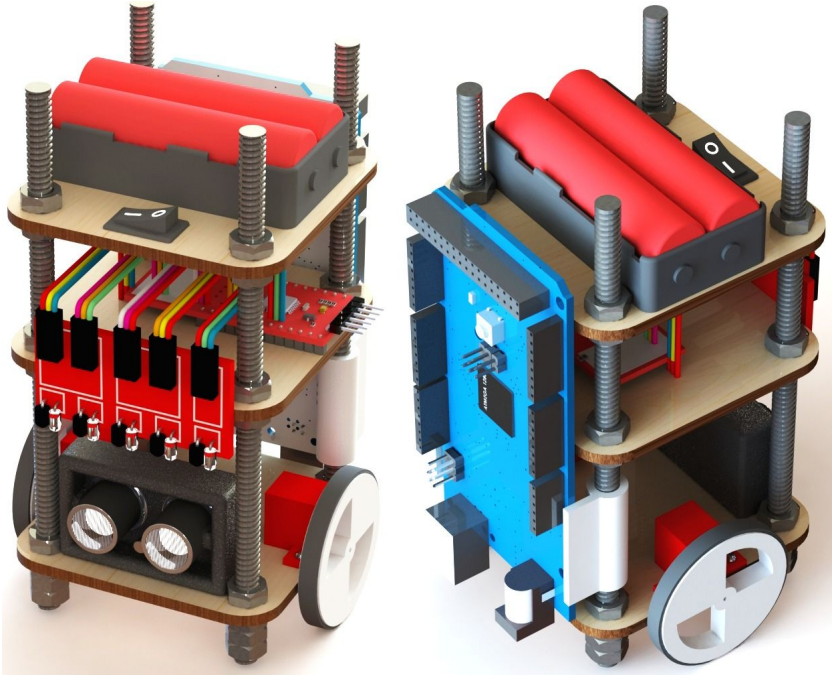
de testar sua funcionalidade em situações de diferentes observações, estruturar os fundamentos para o desenvolvimento dos modelos de indústria 4.0, disponibilizando o código e a lógica para que tal aplicação possa ser usada por outros interessados, permitindo e motivando a modernização e utilização da robótica móvel de maneira intuitiva e simples, como poderá ser visto no capítulo 4.

Figura 3: Fotografia detalhando um robô móvel antes e após a montagem dos componentes eletrônicos.



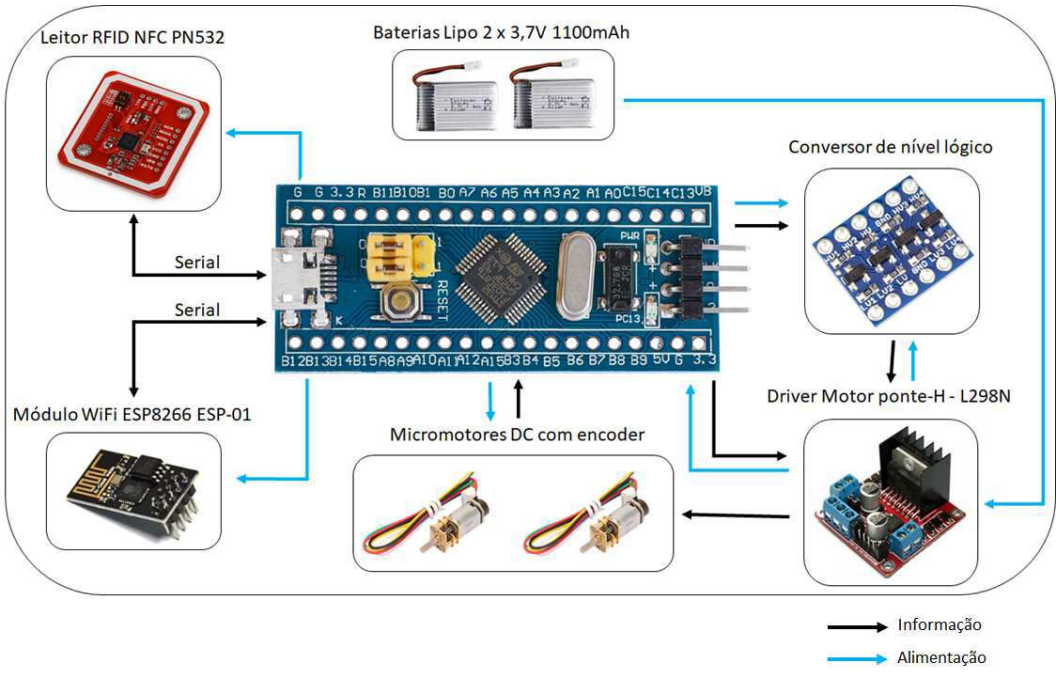
Fonte: Fonseca (2018).

Figura 4: Modelo do segundo robô utilizado.



Fonte: Cortesia de Ferreira (2018).

Figura 5: Representação esquemática dos componentes eletrônicos aplicados aos robôs móveis



Fonte: Fonseca (2018).

1.3. Objetivos

Os objetivos centrais deste trabalho são:

1) Simular um ambiente de armazenamento autônomo e modular

2) Desenvolver uma *Application Programming Interface* (API) para controle dos robôs alocados no armazém

3) Testar ambiente e API com dois robôs previamente desenvolvidos.

4) Divulgar Funcionalidades do *Virtual Robotics Experimentation Platform* (V-REP) em ambiente acadêmico

1.4. Estrutura

O trabalho busca demonstrar o processo de desenvolvimento da solução, primeiramente estruturando a base teórica para o entendimento do projeto , explicando os softwares e algoritmos utilizados. No capítulo 3 serão descritos os materiais e métodos utilizados durante o desenvolvimento do projeto. No capítulo 4 será exposto o desenvolvimento do software juntamente com a evolução das metas definidas por etapa, uma descrição da lógica utilizada e a descrição da funcionalidade da API desenvolvida. No capítulo 5 os resultados encontrados serão descritos e finalizando no capítulo 6 com a conclusão e sugestões para trabalhos futuros. No apêndice são apresentados os códigos de software desenvolvidos, e em anexo de descrições técnicas sobre o funcionamento da interface de comunicação entre a API remota desenvolvida e o ambiente de simulação do armazém.

2 FUNDAMENTAÇÃO TEÓRICA

Uma descrição estrutural desse projeto pode ser definida como: uma simulação de um robô móvel no software V-REP , que tem como sua principal linguagem de programação LUA (algumas interfaces que utilizam C e C++) e controlado através de uma API remota desenvolvida em Python.

Descreve-se neste capítulo os conhecimentos necessários e utilizados para o desenvolvimento deste projeto, abordando-se conceitos como robótica móvel, paradigmas de programação, API de controle externo, além da exposição dos softwares e das linguagens utilizadas no desenvolvimento do mesmo.

2.1 Automação

Roggia e Fuentes (2016) observaram que desde a pré-história o homem vem desenvolvendo mecanismos e invenções com o intuito de reduzir o esforço físico e auxiliar na realização de atividades. A automação de um processo tenta colocar qualquer sistema cada vez mais independente do homem, realizando medições e as correções necessárias de maneira independente. No conceito de automação, ainda de acordo com Roggia e Fuentes (2016), subdividem-se três níveis:

1. Automação rígida: está baseada em uma linha de produção projetada para a fabricação de um produto específico. Apresenta altas taxas de produção e inflexibilidade do equipamento na acomodação da variedade de produção.
2. Automação programável: o equipamento de produção é projetado com a capacidade de modificar a sequência de operações de modo a acomodar diferentes configurações de produtos, sendo controlado por um programa que é interpretado pelo sistema. Diferentes programas podem ser utilizados para fabricar novos produtos. Esse tipo de automação é utilizado quando o volume de produção de cada item é baixo.
3. Automação flexível: reúne algumas das características da automação rígida e outras da

automação programável. O equipamento deve ser programado para produzir uma variedade de produtos com algumas características ou configurações diferentes, mas a variedade dessas características é nor-malmente mais limitada que aquela permitida pela automação programável.

2.2 Robótica móvel

O estudo de robótica móvel geralmente é dividido em dois níveis, primeiro trabalha-se com o robô móvel em um ambiente estruturado, onde se cria um ambiente do qual há o conhecimento prévio ou que seja capaz controlar tudo aquilo que circunda o robô, tal ambiente é chamado de envelope por Luciano Floridi em *The Fourth Revolution* (apud, Átila Amarinho (2019)). No nível 2 da robótica móvel assume-se o robô em ambiente não estruturado, sendo possível a interferência de variáveis de comportamento não determinístico como animais, irregularidades do solo, variações no clima, sendo esse o nível pesquisado no desenvolvimento de carros autônomos, por ser inviável economicamente envelopar todas os estradas do país, isso é isolar uma estrada de agentes não-determinísticos como pessoas ou animais.

Siegwart e Nourbakhsh (2004, apud Fonseca (2018)) destacam que alguns mecanismos de locomoção de robôs são inspirados na locomoção biológica como andar, pular, correr, deslizar, patinar, nadar, voar ou rolar. No entanto o mecanismo de locomoção mais utilizado ainda é a roda tendo o melhor desempenho terrenos planos.

Além de sensores de visão, foi utilizado para localização da posição no armazém a tecnologia NFC, *Near Field Communication* (Comunicação por Campo de Próximo), que como citado por Coskun, Ok e Ozdenizci (2011, apud Fonseca (2018)) “é uma tecnologia emergente e promissora que terá um enorme impacto no ecossistema financeiro bem como na tecnologia móvel de todo mundo”.

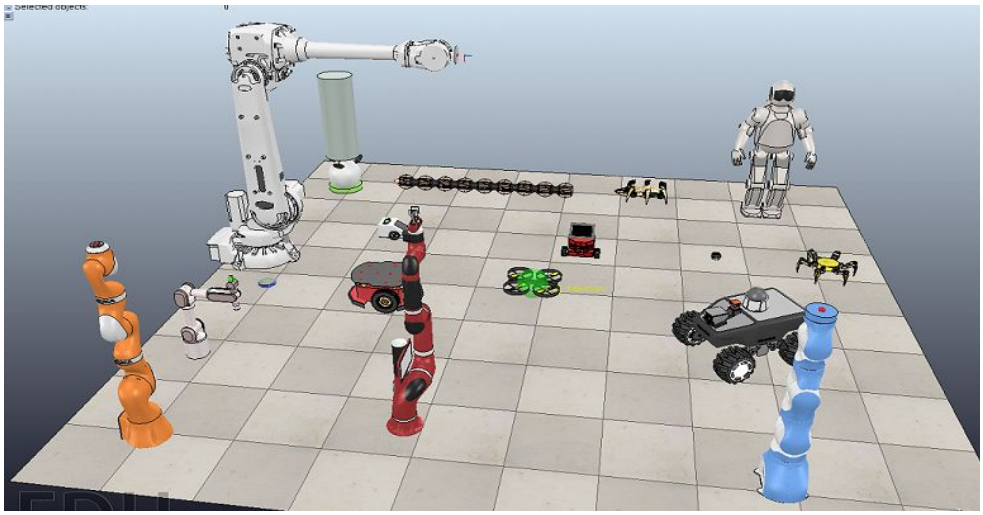
2.3 Simulação em robótica móvel

Como no trabalho realizado por Inam (2018), simulação se mostra importante nos primeiros testes, para orientações gerais em relação ao projeto e em calcular probabilidade de risco ao robô e a humanos.

De acordo com Peralta et al. (2016, apud Fonseca (2018)), no campo da robótica os simuladores são de maior importância porque robôs de alta performance são caros e normalmente escassos nos ambientes de pesquisa. Isto gera competitividade de recursos nas ocasiões em que os pesquisadores desejam realizar seus testes e experimentos ao mesmo tempo.

Entre os simuladores existentes o Virtual Robotics Experimentation Platform (V-REP) (ROHMER et al., 2013) se destaca como um software gratuito para simulação robótica em um ambiente de 3 dimensões. O mesmo possui diversos cenários contendo demonstrações, tutoriais e modelos de robôs, móveis e fixos desenvolvidos de maneira física e ali simulados, como pode ser visto na figura 6, está contêm um vasto exemplo de robôs comerciais que enriquecem e demonstram a versatilidade do V-REP, como por exemplo o Asti (robô humanoide), ant-hexapod e hexapod (os dois robôs com seis pernas), Quadricopter (drone ao centro da figura) e vários robôs manipuladores.

Figura 6: Exemplo de modelos de simulação de diversos robôs disponíveis no V-REP.



Fonte: Elaborado pelo proprio autor.

2.4 Programação

Sendo substituído um ambiente real pelo ambiente simulado, substituições materiais e suas propriedade por linhas de código, buscando encontrar como objetivo uma simulação mais perto do possível da realidade. Descrevem-se as ferramentas deste ambiente, para se alcançar o objetivo.

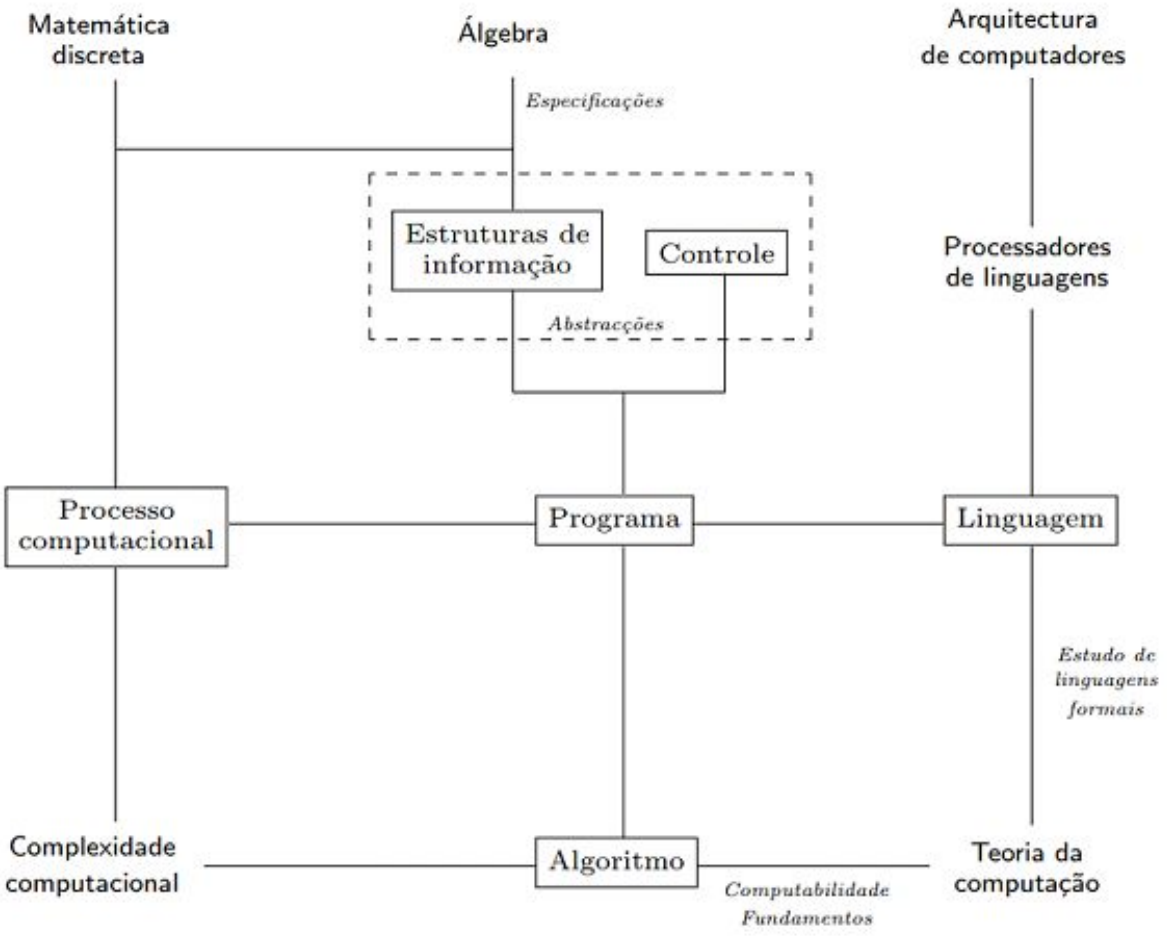
2.4.1 Paradigma de programação

Um programa é uma entidade estática que utilizando uma determinada linguagem origina e define a evolução de um processo computacional (Martins, 2012). Para desenvolver um programa um programador utiliza de um paradigma, esse é um meio de se classificar as linguagens de programação baseado em suas funcionalidades. As linguagens podem ser classificadas em vários paradigmas.Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa. De acordo com Jungthon e Goulart (2010), cada paradigma determina uma forma particular de abordar os problemas e de formular respectivas soluções, e uma linguagem de programação pode combinar dois ou mais paradigmas para potencializar as análises e soluções. Deste modo, cabe ao programador escolher o paradigma mais adequado para analisar e resolver cada problema.

Uma forma de entender o funcionamento e a identidade do que é um programa é ilustrado na figura 7, onde se observa um conjunto de conceitos associados, por Martins (2012) a programa computacional. Um programa engloba abstrai do mundo real através estruturas de informação e controle. Matematicamente programa pode ser interpretado como um algoritmo, isto é uma sequência de passos lógicos para se atingir um objetivo, esses passos são interpretados pelo *hardware* através de uma linguagem de programação, que pode estar mais perto da linguagem da máquina como *assembly* ou mais perto da linguagem humana definidas como linguagens de alto nível, como Python, que depois será

interpretado por um processador de linguagens e passado para o computador de acordo com sua arquitetura.

Figura 7: Mapa dos principais conceitos associados a programa.



Fonte: Martins (2012)

A classificação de um paradigma se dá através de seu conceito de base, podendo se tomar como exemplo esses: Imperativo, funcional, lógico, orientado a objetos e estruturado. A programação orientada a objetos, segundo Jungthon (2010), é baseada na composição e interação de diversas unidades de softwares denominados objetos, sendo estes, módulos do programa. A alteração deste objeto não incorre na modificação de outros objetos; quanto

mais ele for independente, maior a chance dele poder ser reutilizado em outra aplicação. O funcionamento de um software orientado a objetos se dá através do relacionamento e troca de mensagens entre esses objetos. Esses objetos são classes, e nessas classes os comportamentos são chamados de métodos e os estados possíveis da classe são chamados de atributos. Nos métodos e nos atributos também são definidas as formas de relacionamento com outros objetos.

É de conhecimento acadêmico e bem explicado por Gasparotto (2014) que para uma linguagem se encaixar no paradigma de orientada a objeto, classificação que pode ser dada ao V-REP, ela deve atender pelo menos quatro tópicos, tipicamente conhecido como HEAP:

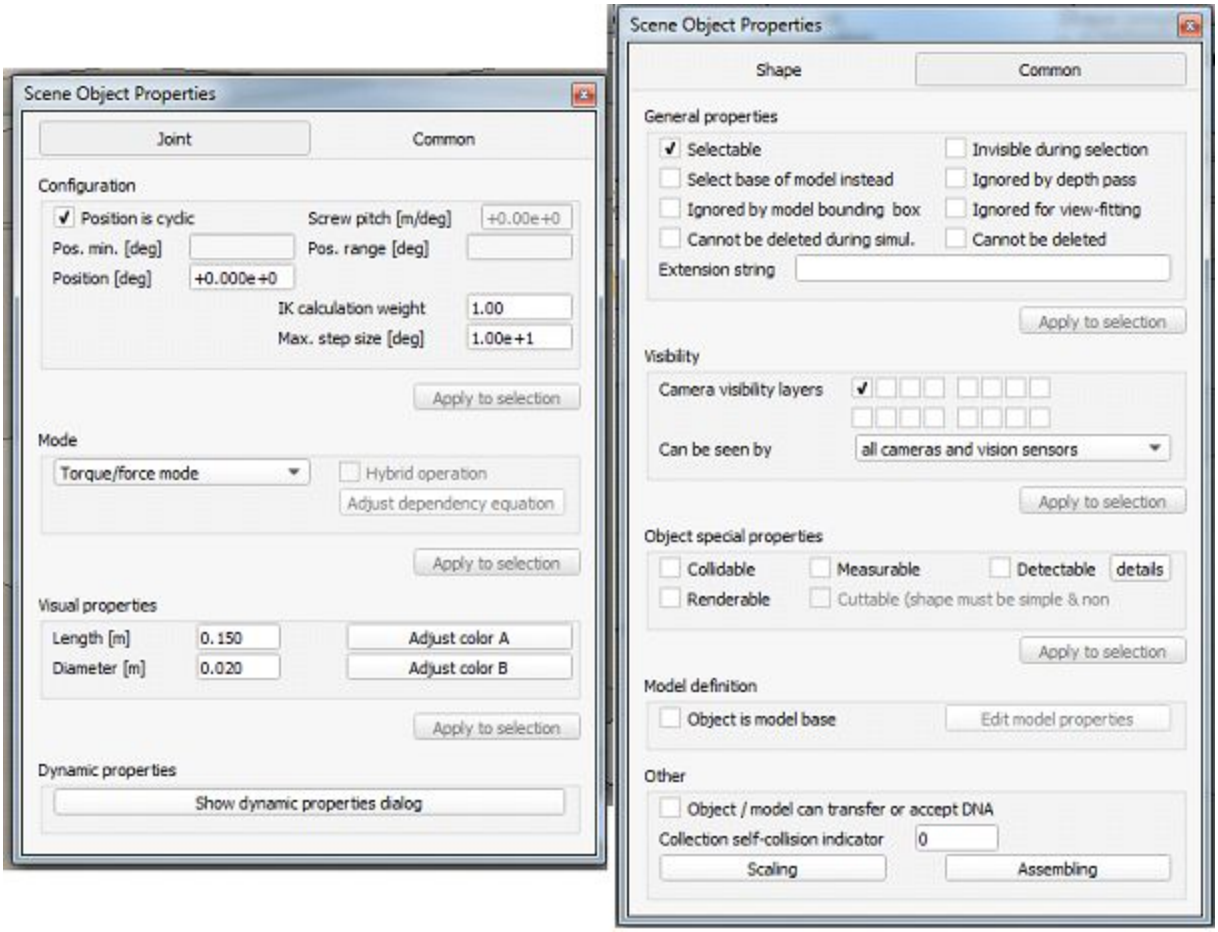
Herança fazendo que ao classificar um objeto como filho de outro ele herde suas características, permitindo o reuso de código. Como visto na figura 8, parte direita, onde independente do item que se cria, identifica-se um conjunto de características comuns a todos objetos, assim todo objeto criado em cena herda da classe objeto comum.

Encapsulamento impede o acesso direto à uma variável, na maioria das linguagens colocam encapsulamento como uma medida de segurança, tanto estrutural como semântica. Declara-se uma variável privada limitando alterações apenas aquelas pré-definidas pelo programador. Tal característica, por exemplo, é vista nas janelas de propriedade da figura 8, onde o conjunto de alterações do objeto são permitidas apenas nos limites colocados pelos desenvolvedores.

Abstração é aquilo que dá identidade ao objeto, definindo características (propriedade) e funções (métodos) que aquele objeto pode realizar. Como já foi observado, a parte comum que foi herdada na abstração se observa através das configurações que distingue um objeto em cena do outro.

Polimorfismo é a possibilidade de usar o mesmo nome para métodos diferentes, como o método que muda cor de uma junta, que possui duas cores, ou de um objeto cilíndrico, que possui uma cor apenas, ambas se classificam com o mesmo nome.

Figura 8: Caracteristicas comuns e especificas de objetos no V-REP.



Fonte: Elaborado pelo proprio autor.

Mesmo com estes conceitos estabelecidos, é esclarecido por Biondo (2017) “paradigmas não são isolados uns dos outros; devido a uma mescla de conceitos que formam os paradigmas, uma linguagem de programação pode atender a um ou mais paradigmas”.

2.4.2 API externa

Um princípio fundamental para muitas tecnologias existentes hoje é a comunicação, tendo a área de internet das coisas, ou do inglês internet of things (IoT) o mais claro exemplo disto e se mostrando cada vez mais presente no cotidiano das pessoas. De acordo com Free On-line Dictionary of Computing (FOLDOC) (1995) *Application Programming Interface* (API), em tradução para o português "Interface de Programação de Aplicativos", é “a interface pela qual um programa acessa o sistema operacional e outros serviços.”

Uma API é criada quando uma empresa de software tem a intenção de que outros desenvolvedores de software elaborem produtos associados ao seu serviço. Existem vários deles que disponibilizam seus códigos e instruções para serem usados em outros sites da maneira mais conveniente para seus usuários, respeitando regras de utilização de compartilhamento. Este trabalho faz o uso de API desenvolvida durante o projeto estrutura-se com a base do cliente-servidor tendo o servidor disponibilizando os recursos para o programa em Python.

2.5 Software e Linguagens usadas

Neste tópico serão descritos os componentes utilizados no desenvolver do projeto, baseando-se na utilização de softwares livres e sistemas com licença gratuita, possibilitando o teste, a reanálise e a extensão deste projeto sem maiores custos. Os conceitos descrito na primeira parte deste tópico focam nos recursos utilizados para implementação, sendo eles V-REP, linguagem de programação Python e Lua (sendo essa a linguagem adotada para programação de scripts internos do V-REP).

2.5.1 V-REP

De acordo com Coppelia Robotics (2019a), o simulador robótico V-REP, com ambiente de desenvolvimento integrado, é baseado em uma arquitetura de controle distribuída: cada objeto / modelo pode ser controlado individualmente através de um script embutido, um plugin, um nó ROS ou BlueZero, um cliente API remoto ou uma solução personalizada. Isso torna o V-REP muito versátil e ideal para aplicações com vários robôs. Controladores podem ser escritos em C / C ++, Python, Java, Lua, Matlab ou Octave.

Segundo Coppelia Robotics (2019b), a API remota do V-REP em Python é composta por aproximadamente cem funções específicas e uma função genérica, as funções utilizadas no desenvolver do trabalho são descritas na metodologia. As funções API remotas interagem com o V-REP, de forma oculta para o usuário, via comunicação de socket (ou, opcionalmente, via memória compartilhada), permitindo que um ou vários aplicativos externos interajam com o V-REP de forma síncrona ou assíncrona, suportando inclusive o controle remoto do simulador (carregando uma cena remotamente, iniciando ou parando uma simulação por exemplo).

2.5.2 Lua

Lua.org (2018) define que Lua é uma linguagem de programação poderosa, eficiente e leve, projetada para estender aplicações. Ela permite programação procedural, programação orientada a objetos, programação funcional, programação orientada a dados e descrição de dados.

Seus fundadores ainda posicionam Lua como linguagem ideal para automação (*scripting*) e prototipagem rápida. Isto se dá pois a linguagem combina “sintaxe procedural simples com poderosas construções para descrição de dados baseadas em tabelas associativas e semântica extensível.”

Lua é tipada dinamicamente, isto é suas variáveis possuem uma classificação sobre aquilo que armazenam e realizam facilmente a troca entre suas classificações por exemplo

de inteiro para *float*, é executada via interpretação de *bytecodes* para uma máquina virtual baseada em registradores, e tem gerenciamento automático de memória com coleta de lixo incremental (LUA.ORG, 2018).

2.5.3 Python

Destaca-se para necessidade desse projeto a compreensão que Python, mesmo sem se limitar ao paradigma, pode ser descrito como uma programação orientada a objeto, que orienta o V-REP, fazendo com que a API externa desenvolvida tenha semelhanças muito fortes com código original, e fortalecendo que sua programação se torne bastante intuitiva.

Em entrevista oferecida para Artima (2003), Guido van Rossum (criador da linguagem) afirmou que ‘’Python é uma linguagem de script de alto nível. De outra perspectiva, pode-se dizer que é uma linguagem de programação de alto nível que é implementada de uma forma que enfatiza a interatividade.’’ Rossum comenta ainda sobre sua tipificação fraca e sua extensibilidade focando já naquela época usar a linguagem em diversas plataformas.

Atualmente a linguagem é gerenciada pela *Python software Foundation*, empresa sem fins lucrativos e baseada no modelo de desenvolvimento comunitário, permitindo que vários usuários cooperem para expansão de sua documentação e de suas funções. Em 2018 foi considerada a terceira linguagem mais amada por programadores em uma pesquisa realizada pelo Stack Overflow, um dos principais sites de perguntas e respostas voltado para programadores e entusiastas. Suas principais características, destacadas pela PyScience-Brasil (2019), são sua fácil legibilidade e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens. Observa-se na figura 9 um exemplo de codigo python, onde se declara um função.

Figura 9: Exemplo de codigo em Python

```
def Testesensor(Leitura):    """declara a função """
    global Armazem           """coloca Armazem como variavel global"""
    cor=list(Leitura[2])     """transforma o segundo elemento de leitura em uma lista e armazena"""
    if(len(cor)==1):         """testa o tamanho da lista armazenada"""
      cor=list(cor[0])       """se for cor recebe o primeiro endereço da lista"""
    leu=False                """variavel RECEBE falso"""
```

Fonte: Elaborado pelo próprio autor

3 METODOLOGIA E DESENVOLVIMENTO

Como desenvolver este trabalho visa a utilização do V-REP para simular a estrutura de um armazém cujas decisões do robô serão tomadas via *script* Python, buscou-se abranger a funcionalidade do código, testando-o para dois modelos de robôs móveis, apresentados recentemente por Ferreira et al. (2018) e por Fonseca (2018), ambos inspirados em tecnologia de prototipagem rápida e impressão 3D, e também testando dois meios de identificação de localização e auxílio à navegação, trilha com saturação de cor, permitindo implementar a navegação baseada no seguimento de linha e comunicação por campo próximo (NFC), proporcionando a identificação de posições estratégicas por meio de etiquetas espalhadas no ambiente capazes de comunicar sem fio com um leitor posicionado na base do robô.

Os robôs analisados neste estudo utilizam três sensores de visão para seguir uma linha demarcada e se locomover pelo armazém e sensores ultrassônicos para detectar objetos próximos e evitar colisões. Para se localizar e tomar as decisões sobre navegação, utilizam um sistema *wireless* via antenas, por meio de nove antenas anexadas ao piso do armazém, que transmitem, quando próximas à antena do robô, a posição na qual o mesmo se localiza.

Tendo já estabelecido no capítulo 2 a complexidade que pode estar escondida na área de desenvolvimento de software, engenheiros e programadores estabeleceram métodos para verificar-se o desenvolvimento de uma estrutura não física,como um prédio ou um carro, mas abstrata. Visando aprimorar e fundamentar estas metodologias, Royce (1970) definiu o que hoje é conhecido como modelo cascata, um modelo de processo sequencial no qual o início da próxima etapa só se realiza com o fim da anterior.

Por outro lado, Boehm (1988) definiu como seria o desenvolvimento de um software segundo o modelo espiral, desenvolvimento este iterativo e incremental no qual etapas são descartadas, recomeçadas ou adicionadas de acordo com a necessidade do projeto, reconhecendo um desenvolvimento contínuo. Na parte final, assumiu-se uma construção espiral, com ajustes e adaptações constantes. Por ser uma aplicação com desenvolvimento baseada em softwares livres. O desenvolvimento do trabalho seguiu um desenvolvimento em

cascata, durante as três primeiras etapas, adotando-se a metodologia espiral nas etapas 4, 5 e 6.

1- Familiarização com o software de base e seu ambiente,

2- Estruturação dos robôs no ambiente de simulação,

3- Adaptação dos scripts dos robôs para integração com API remota

4- Desenvolvimento do *Script* em Lua anexado ao robô para deslocamento em linha,

5- Desenvolvimento da API externa para deslocamento em linha,

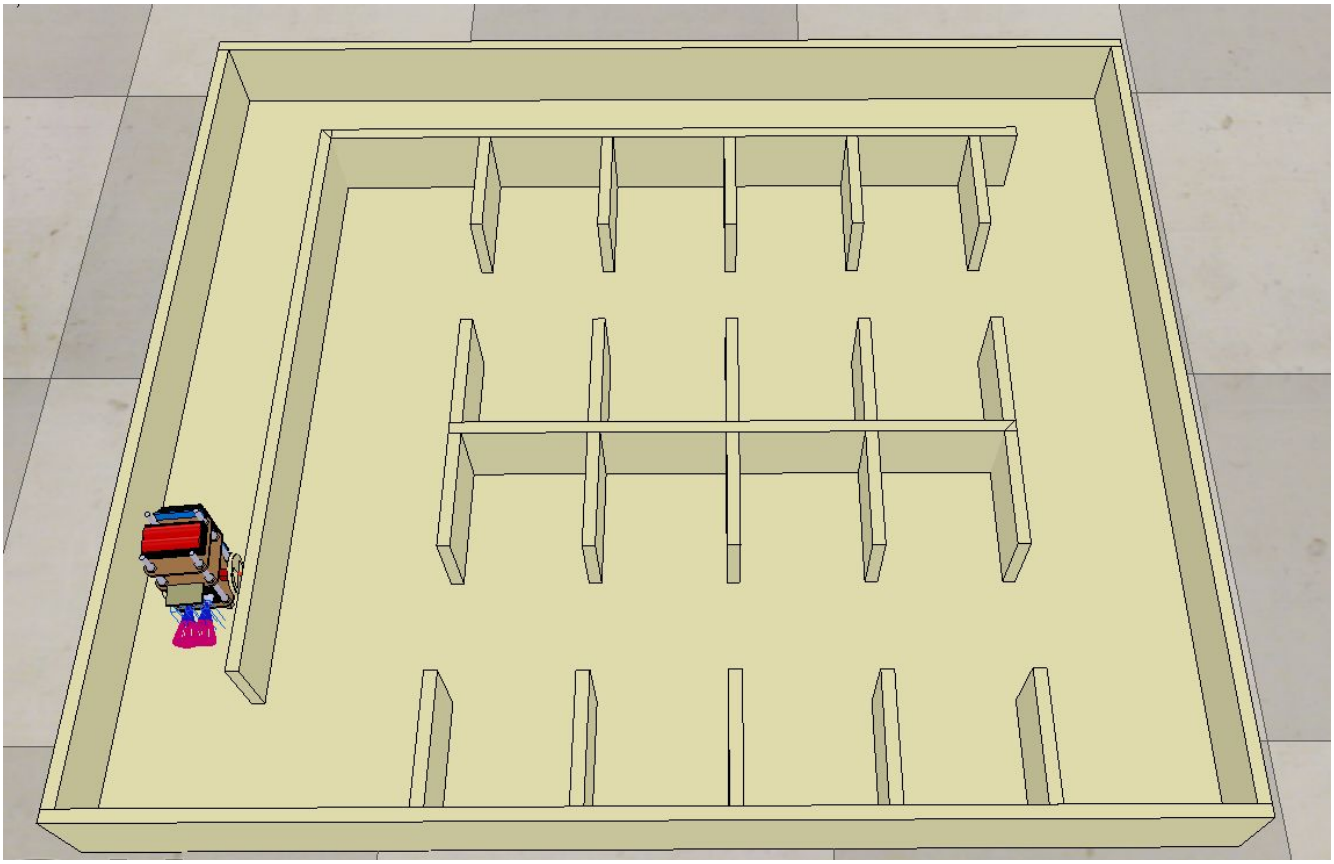
6- Adaptações na API externa para o cenário do armazém.

3.1 Análise do problema

No desenvolver de um mecanismo de controle para um armazém autônomo, optou-se por desenvolver o ambiente de teste de maneira simulada. Tal escolha possibilita o compartilhamento não só dos resultados como também do ambiente de teste, junto com o fato que minimizará os custos relacionados ao projeto.

Observa-se na figura 10 o ambiente simulado do armazém, composto por dezesseis células de armazenamento internas e três corredores que circundam a região de armazenagem, no qual o robô deverá se deslocar de forma autônoma. Como neste caso específico a largura dos corredores não permite a passagem de robôs em paralelo, considerou-se que qualquer corredor do armazém será de mão única e o percurso do robô cíclico. Para definição de cenários, considerou-se o ponto de partida do robô como o corredor lateral esquerdo, movimentando-se em direção à parte inferior, como indicado na figura 10.

Figura 10: Bancada preparada para a simulação de um armazém.



Fonte: Cortesia de Ferreira (2018)

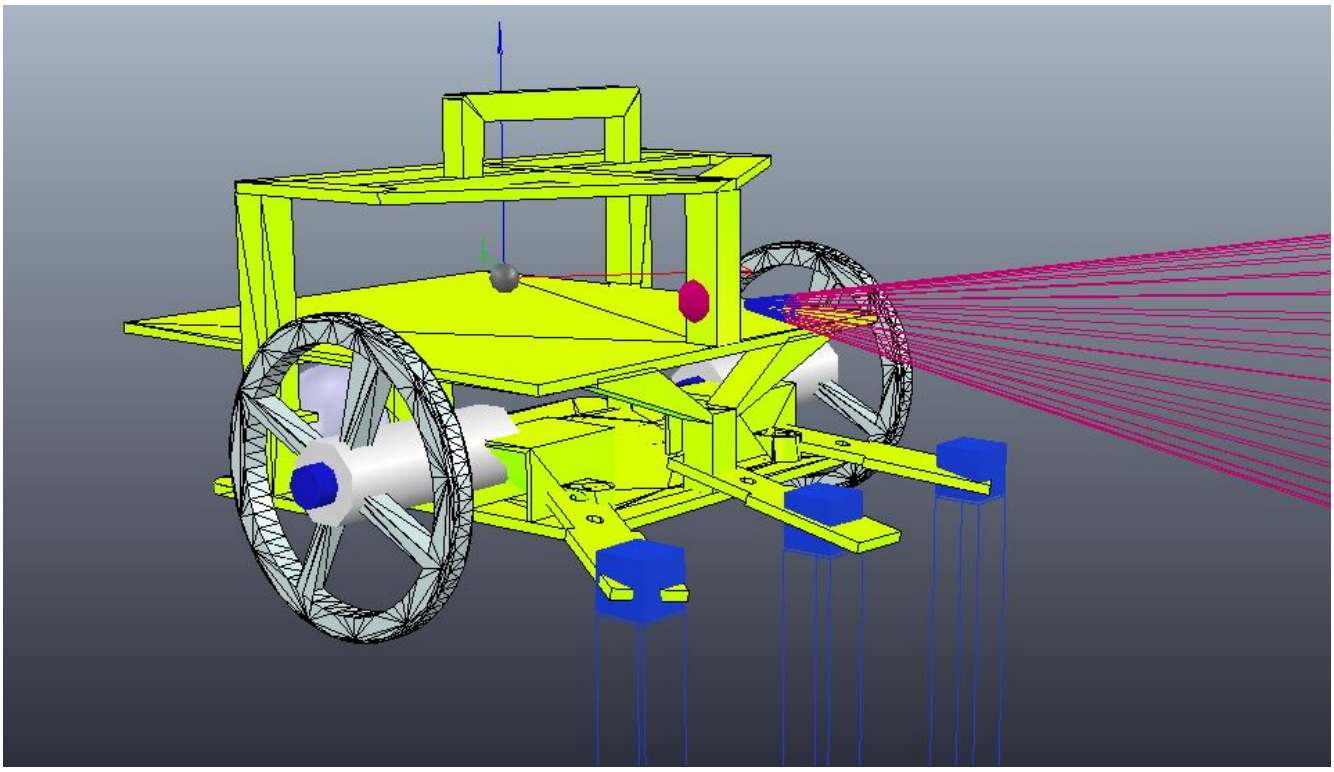
3.2 Robôs

Para uma melhor compreensão da dinâmica e das funções do V-REP, seguiu-se o tutorial didático BubbleRob (COPPELIA ROBOTICS, 2019c) buscando implementar a mesma lógica de operação aos modelos de simulação dos robôs considerados neste estudo.

O denominado durante o trabalho de TR1, cortesia de Fonseca (2018), cujo chassi em ambiente de simulação pode ser observado na figura 3, durante a execução o robô foi denominado TR1, foi adaptado adicionando-se um suporte para os sensores e otimizações em relação à dinâmica e reduções na complexidade de imagens para facilitar o processamento em tempo de execução, isto se realizou através da redução do número de

triângulos que formam superfície do objetos de cena. Tal função disponibilizada pelo V-REP é conhecida como *decimate selected shapes.* Na sequência, foram embutidos ao Chassi os três sensores para o seguidor de linha, juntamente com seus apoios, e um sensor ultra-sônico, como mostrado na Figura 11.

Figura 11 : Modelo modificado do primeiro robô utilizado, denominado TR1.



Fonte: Modificado de Fonseca (2018).

O Robô 2, cortesia de Ferreira et al. (2018), cujo modelo de simulação pode ser visualizado na figura 4, recebeu adaptações no posicionamento de antenas para comunicação *wireless* e configurações nas rodas para maior aderência ao terreno do armazém. Para ambos os modelos abordados, buscou-se manter as proporções e as posições de cada uma das peças, mantendo todas adaptações necessárias a nível de ambiente e algoritmo.

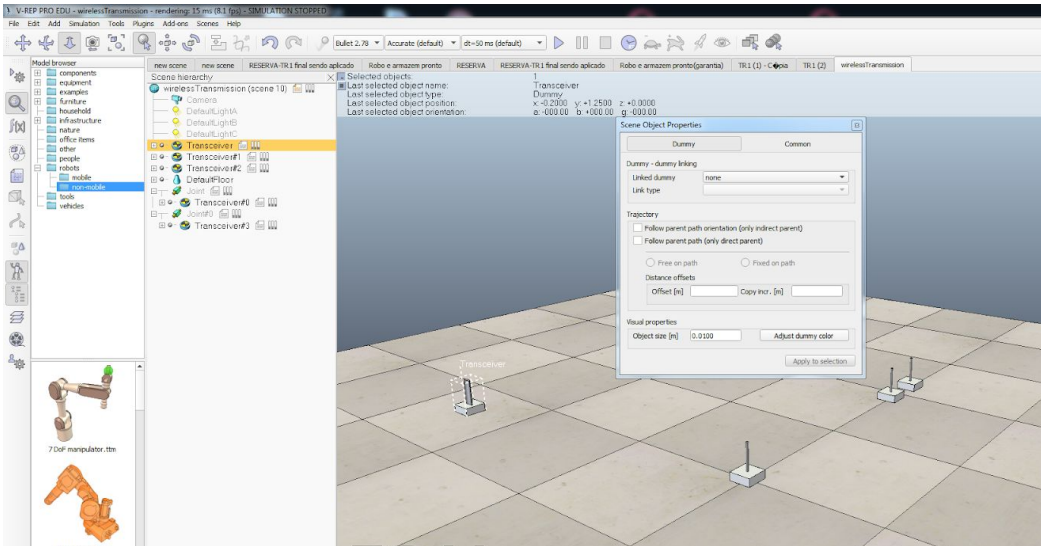
3.3 Antenas NFC

A comunicação NFC no ambiente de simulação do V-REP foi implementada por meio de objeto de *dummy*, objeto este demonstrado na Figura 12.

De acordo com Coppelia Robotics (2019d), um objeto fictício, denominado *dummy*, é o objeto mais simples disponível: é um ponto com orientação e pode ser visto como um quadro de referência. Os autores destacam que *dummies* são objetos multiuso que quando usados em conjunto com outros objetos ou módulos de cálculo , eles podem ser de importância crucial.

Um *dummy* é um recurso bastante amplo do V-REP, ao qual pode se associar um *script* em Lua, anexado na seção de códigos, para desempenhar o papel da antena *wireless*. Sua comunicação com a API remota se dá através do envio de mensagem, de modo que no *script* do próprio d*ummy* é definida uma S*tring* que muda de valor toda vez que é lida. Suas configurações podem ser vistas na janela da figura 12. Por questão de estética, nos modelos finais os objetos *dummies* foram configurados como invisíveis.

Figura 12: Cenário com modelos de Dummies associados a antenas *wireless*.



Fonte: Coppelia Robotics (2018).

3.4 API remota

Uma API remota é utilizada para controlar um elemento do V-REP, como um robô, a partir de um processo externo. Isto permite controlar robôs de diferentes ambientes (Windows, ROS etc.). Pode-se, por exemplo, criar um nó no ROS ou um programa ".exe" no Windows ou Linux. Para uma melhor compreensão do funcionamento desta interface e dos programas desenvolvidos, descreve-se na tabela 1 os principais comandos usados na elaboração do código, tanto em Lua, como em Python para a programação da API, os quais estão melhor detalhados em Coppelia Robotics (2019e).

O processo de comunicação cliente-servidor dos comandos relacionados a comunicação entre o script Python e o software V-REP está indicado no Anexo 1.

Tabela 1. Comparação de comandos em LUA e Python .

V-REP (LUA)	Python	Descrição
sim.setJointTargetVelocity	vrep.simxSetJointTargetVelocity	Altera a velocidade dos motores do robô .
sim.readProximitySensor sim.readVisionSensor	vrep.simxReadVisionSensor vrep.simxReadProximitySensor	Recebe dados sobre a leitura dos sensores.
getObjectHandle	vrep.simxGetObjectHandle	Declara para o programa as partes do robô, baseado no nome.
	vrep.simxGetIntegerParameter	Recebe um valor inteiro.
sim.getObjectAssociatedWithScript		Importa a identificação do objeto que está associada ao script
simRemoteApi.start('PortaDeComunicação')	vrep.simxStart (String 'EnderecoIP', PortaDeComunicação, booleanas de controle sobre conexão e reconexão, inteiros sobre velocidade de ciclo de *thread*)	Estabelece a comunicação entre o V-REP e o script em Python, caso a comunicação não se estabeleça é necessário tratar erro.

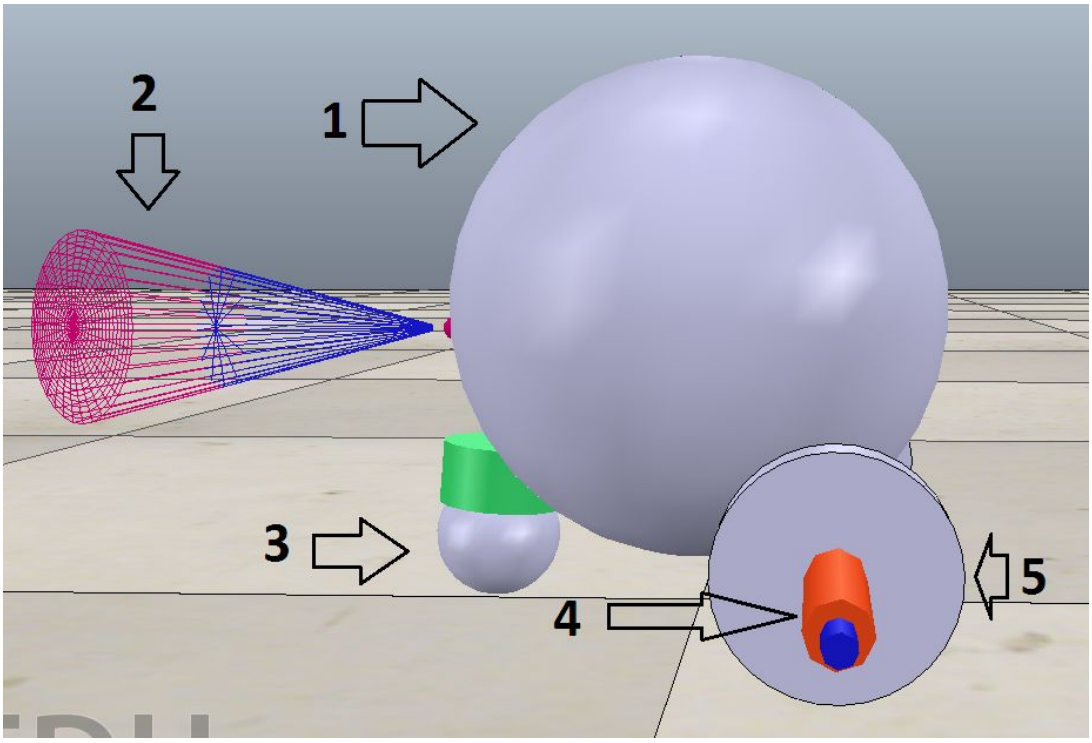
4 IMPLEMENTAÇÃO

Para a verificação da implementação do projeto, foram preparados quatro cenários que simulam um armazém automatizado, com seu controle de decisão funcionando com uma API escrita em python. A implementação deste é descrita nos próximos parágrafos.

4.1 Estruturação do robô

A familiarização com software ocorreu inicialmente através da execução do Tutorial BubbleRob (COPPELIA ROBOTICS,2019c), através do qual foi possível entender a estrutura de comandos dinâmicos e adaptações com com elementos básicos, como *shapes* (blocos sólidos para estruturação da cena), *force sensor* (objetos com a função de permitir a anexação de algum componente ao outro, permitindo deslocamento de um em relação ao outro) preparação das *revolute joints,* motores (que fornecem torque e velocidade angular) e sensores (para percepção do cenário), bem como a estruturação da lógica de operação em *script* Lua. A figura 13 representa o resultado desta etapa do processo, na qual as partes do robô se distinguem em: (1) Chassi; (2) Sensor ultrasônico; (3) Apoio; (4) Motor; (5) Rodas. Dados sobre as dimensões e propriedades das partes robô são encontrados na secção 4.6.

Figura 13: Robô desenvolvido baseado no tutorial *BubbleRob.*



Fonte: Desenvolvido pelo próprio autor.

4.2 Adaptação do robô a receber um script

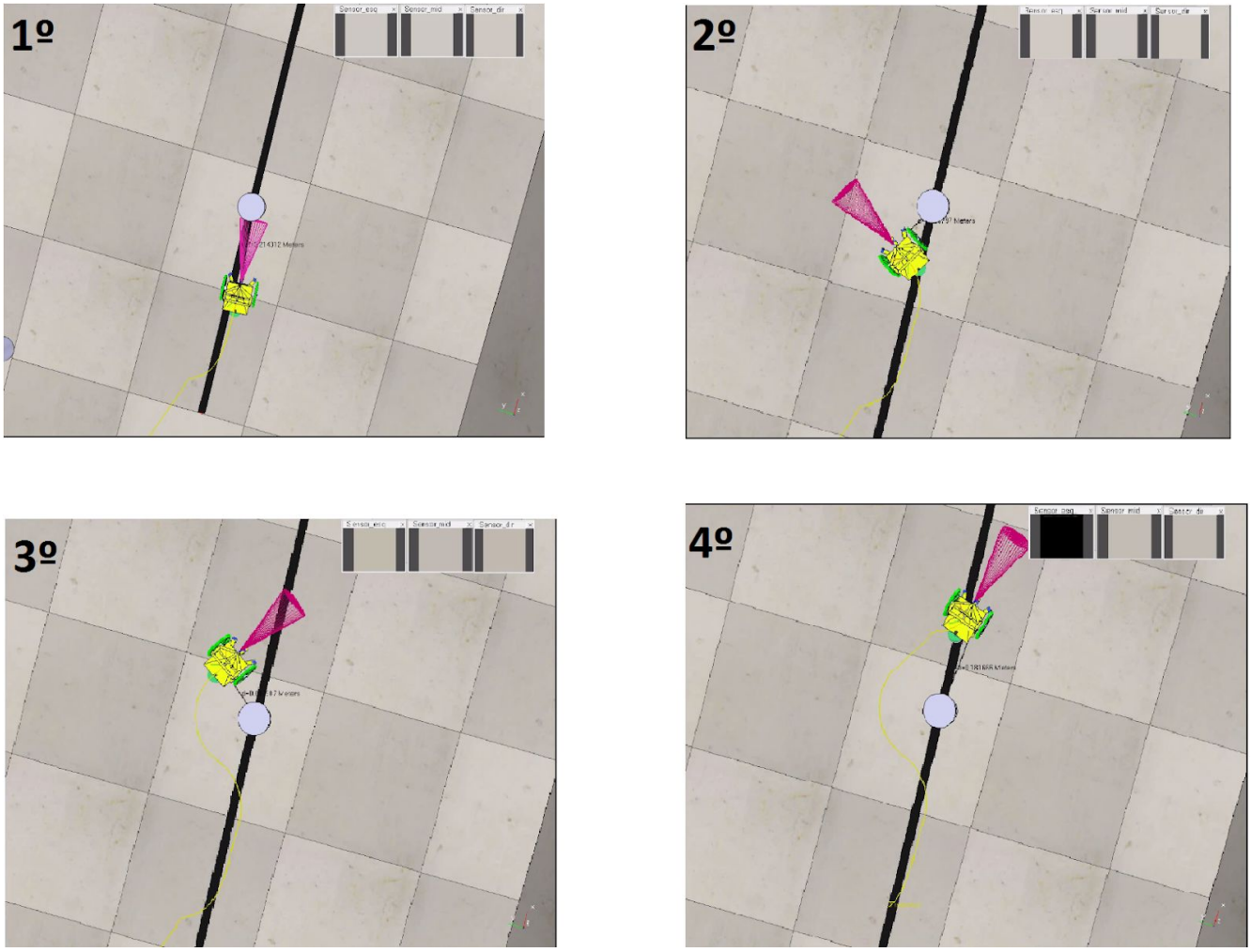
Nesta etapa adaptou-se o TR1, apresentado na figura 11 para percepção tanto do sensor de colisão, quanto para detecção de linha conforme a figura 14 demonstra. Neste caso o objetivo era implementar o comportamento de um seguidor de linha, com habilidade de desviar de obstáculos, e tendo características do robô, como o tamanho e a distância da roda em relação ao chassi, distância dos sensores de linha.

Ainda nesta etapa, utiliza-se a variável de inclinação, a qual, representa a diferença entre as velocidades das duas rodas, quando for necessário realizar alguma curva. Observa-se na figura 14 o resultado dessa etapa com o comportamento do robô móvel desviando de um objeto posicionado sobre o caminho indicado na trilha de cor preta.

4.3 Script em Lua anexado ao robô para deslocamento em linha

Desenvolveu-se um *script* ainda anexado ao software do V-REP, aprimorando-se as variáveis de inclinação, testado-se a capacidade do robô em sair da trajetória estabelecida pela trilha para desviar de um obstáculo e, após isso, retornar para a trilha, culminado na lógica que é descrita na figura 15. Em tal processo passou-se a utilizar o próprio tempo da simulação para executar movimentos ensaiados, tal lógica mais tarde foi usada para que o robô entrasse e saísse dos armazéns.

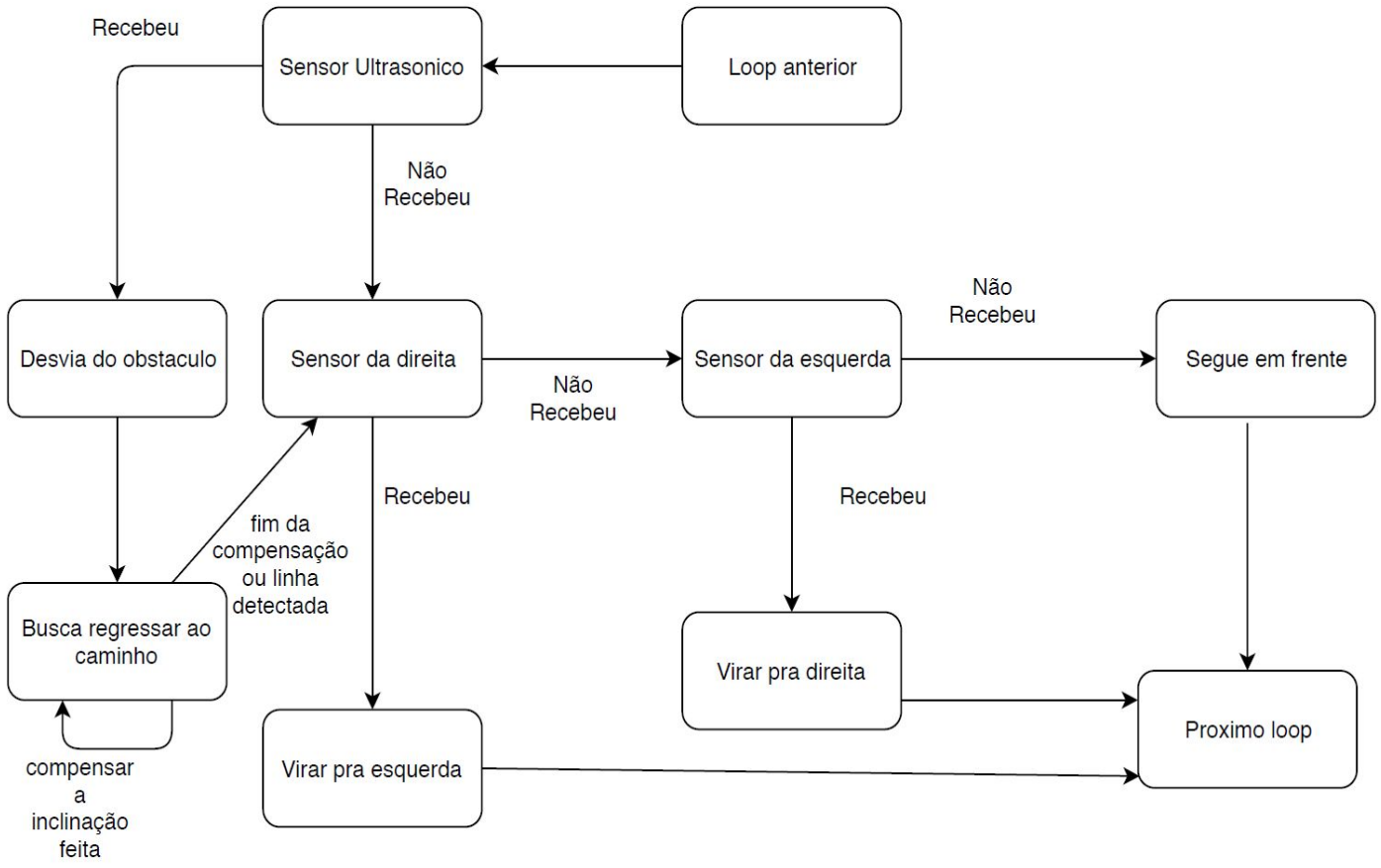
Figura 14: Robô realizando desvio de obstáculo



Fonte: Próprio autor.

Observa-se na figura 13 que a detecção de linha só ocorre caso o sensor ultrassônico não perceba nenhum obstáculo à sua frente, evidenciando o comportamento reativo de desvio de obstáculo.

Figura 15: Lógica de controle primária do robô.

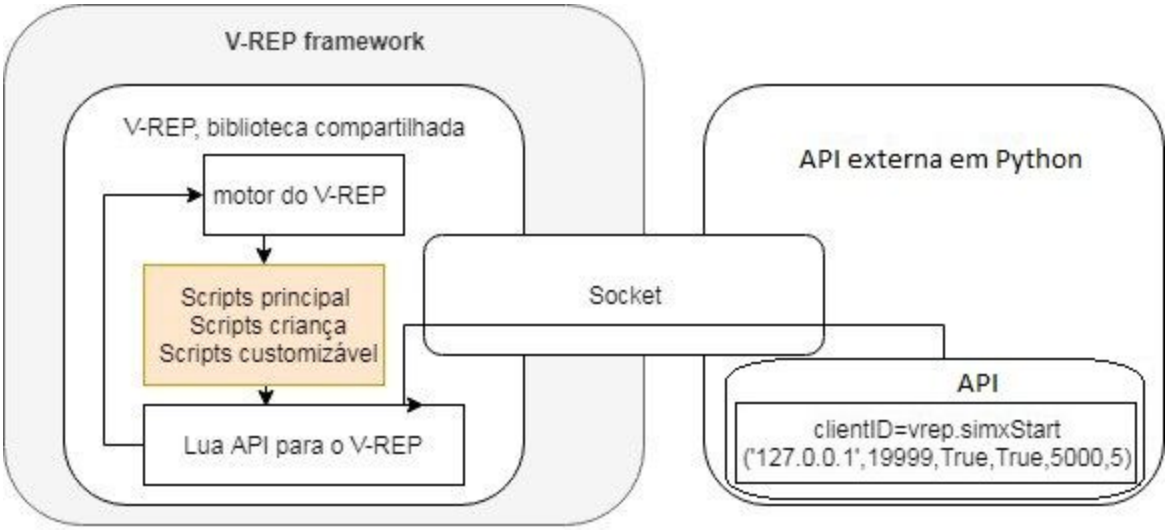


Fonte: Próprio autor.

4.4 API externa para deslocamento em linha

Após a conclusão do script em Lua, buscou-se desenvolver uma API externa em Python. Tal API inicialmente realizava a mesma função do script em Lua, porém, com a implantação em Python foi possível um maior trabalho com os dados, tendo os valores obtidos dos sensores armazenados de maneira mais simples, e uma melhor interface com o usuário. Após ajuste das configurações necessárias para que a conexão via *socket*, demonstrada na figura 16, estivesse estabelecida, configurações extras dependeriam de codificação em Python. Tal interface permite armazenar e gerir as variáveis de um robô utilizando todas as ferramentas que a própria linguagem fornece.

Figura 16: Interação entre V-REP e API externa.



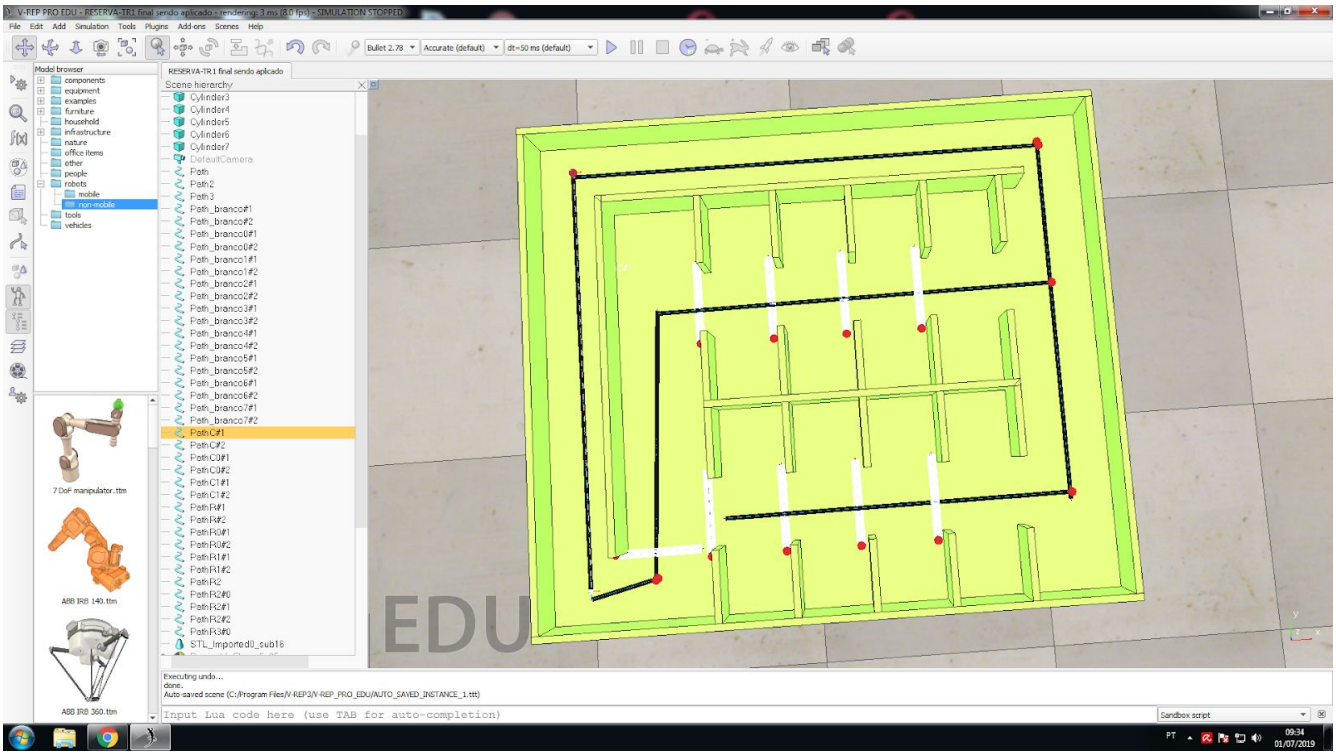
Fonte: Modificado de COPPELIA ROBOTICS (2019f)

4.5 Adaptações da API externa para o cenário do armazém

Após a API comandar ações para que o robô se locomovesse de acordo com o comportamento de seguidor de linha, o objetivo tornou-se fazer com que ele se deslocasse

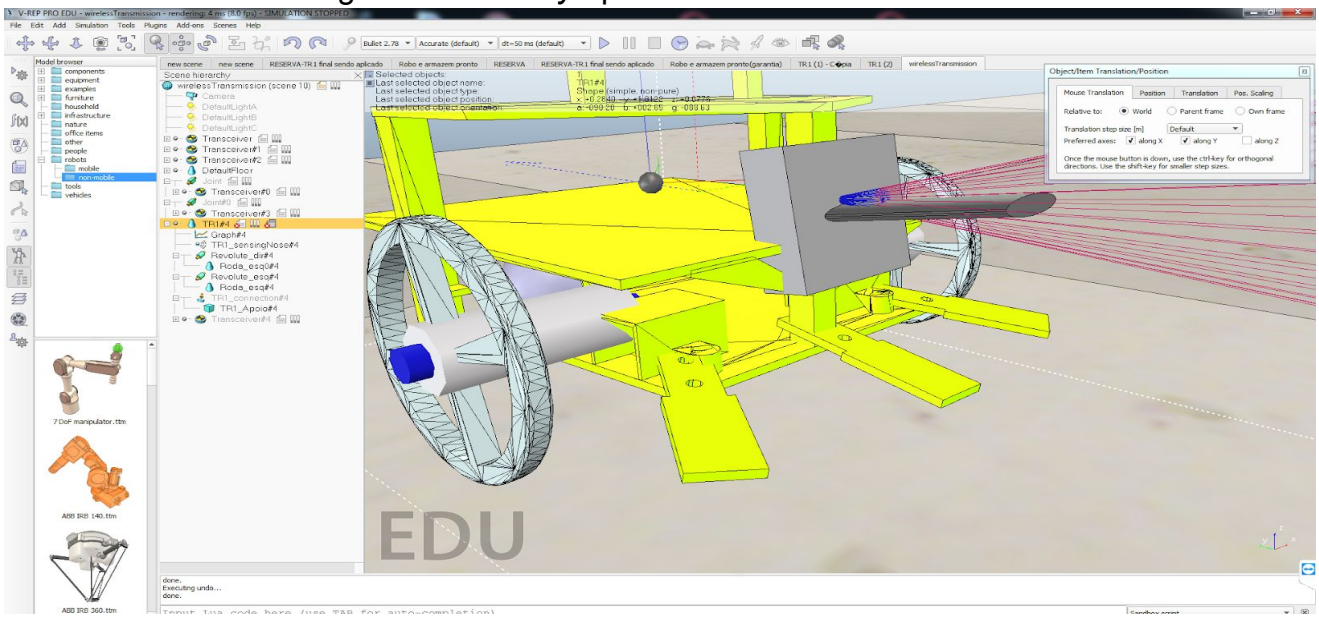
de maneira controlada em uma bancada que simula um armazém com 16 posições de estocagem. Para isso, uma configuração extra foi implementada de maneira a permitir a tomada de decisões. Inicialmente, utilizaram-se linhas brancas, como visto na figura 17, as quais eram codificadas de maneira diferente pelo sensor de visão, permitindo ao robô conhecer qual atitude tomar, baseado na lógica explicitada na figura 19. O segundo método de localização e tomada de decisão baseou-se em um sistema de antenas *wireless* distribuídas em posições estratégicas do armazém e anexado ao robô, como ilustrado na figura 18. Tais antenas não são apenas detectáveis, como são capazes de trocar informações via comunicação de campo próximo, transmitindo dados ao robô e tornando a contagem de linhas que acontecia na API um fator extra para tomada de decisão. Assim como exposto no capítulo 2, a estrutura *dummy* simula um elemento NFC. A implementação do sistema de identificação NFC baseou-se em um cenário pré-projetado, fornecido pela Coppelia Robotics denominado *Wireless Transmission*, e realizaram-se adaptações para o propósito do projeto visto no anexo.

Figura 17: Modelo do armazém com caminhos demarcados por linhas brancas



Fonte: Proprio autor

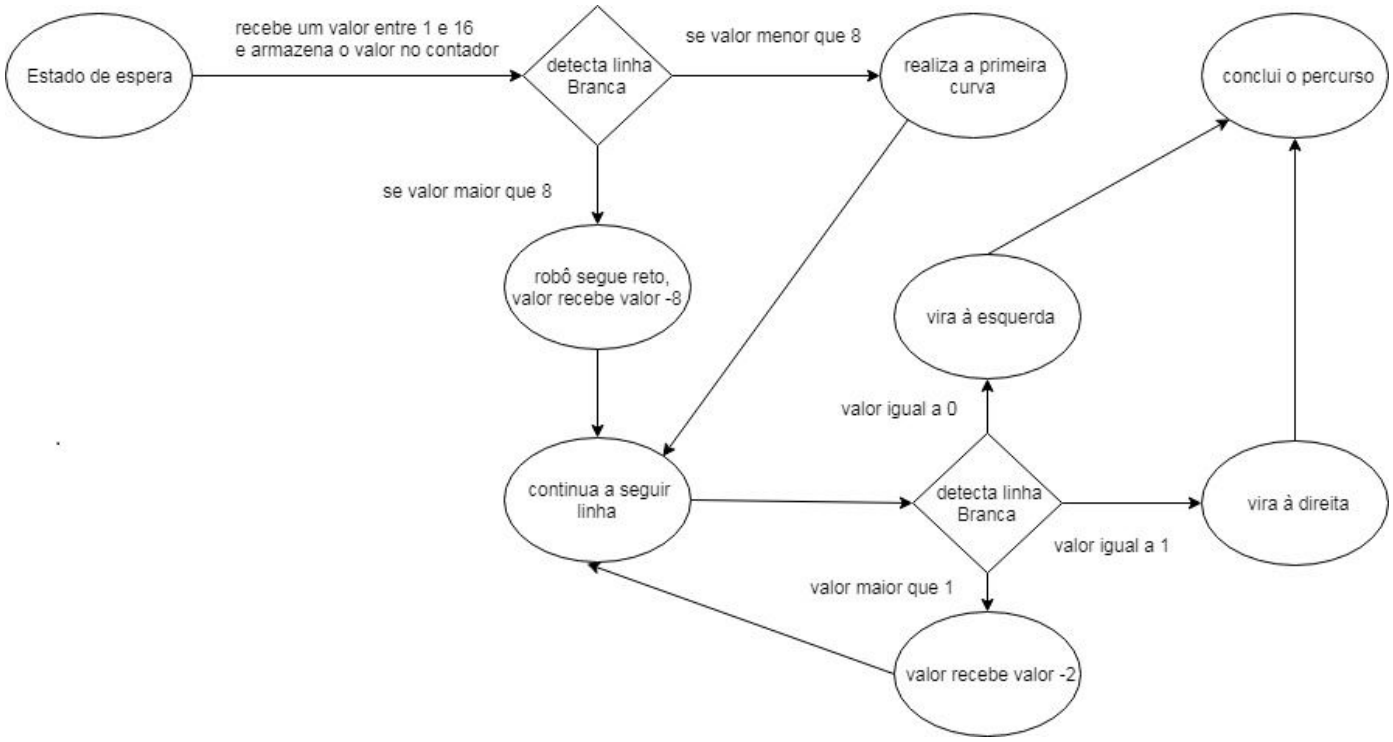
Figura 18: *Dummy* operando como elemento NFC



Fonte: Elaborado pelo próprio

Observa-se na figura 19 o algoritmo desenvolvido em Python para navegação por segmento de linhas brancas, no qual o robô não tem acesso a por qual linha se está passando, logo se utiliza do conhecimento prévio da posição para tomada de decisão. Neste caso utiliza-se uma variável denominada valor, responsável por viabilizar o controle sobre a posição do robô.

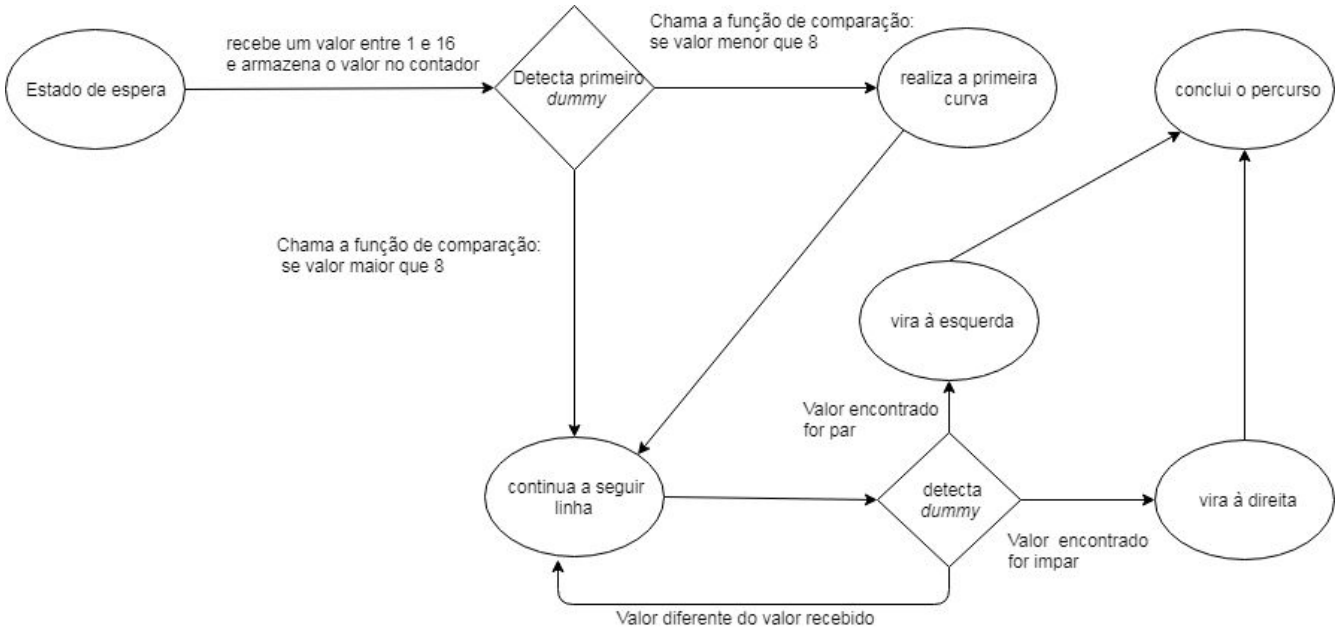
Figura 19: Lógica funcionamento do robô utilizando identificação por linha branca.



Fonte: Elaborado pelo próprio autor

Posteriormente, com interfaces NFC, com as quais o robô consegue interpretar melhor sua posição, pode-se observar uma simplificação entre a figura 19 e a figura 20. Altera-se em relação a lógica anterior que para a primeira curva, utiliza-se um booleano, e como não há redução da constante (que resulta no resultado de 0 ou 1 da figura 19), utiliza-se da verificação do valor da estante ser par (esquerda) ou ímpar (direita).

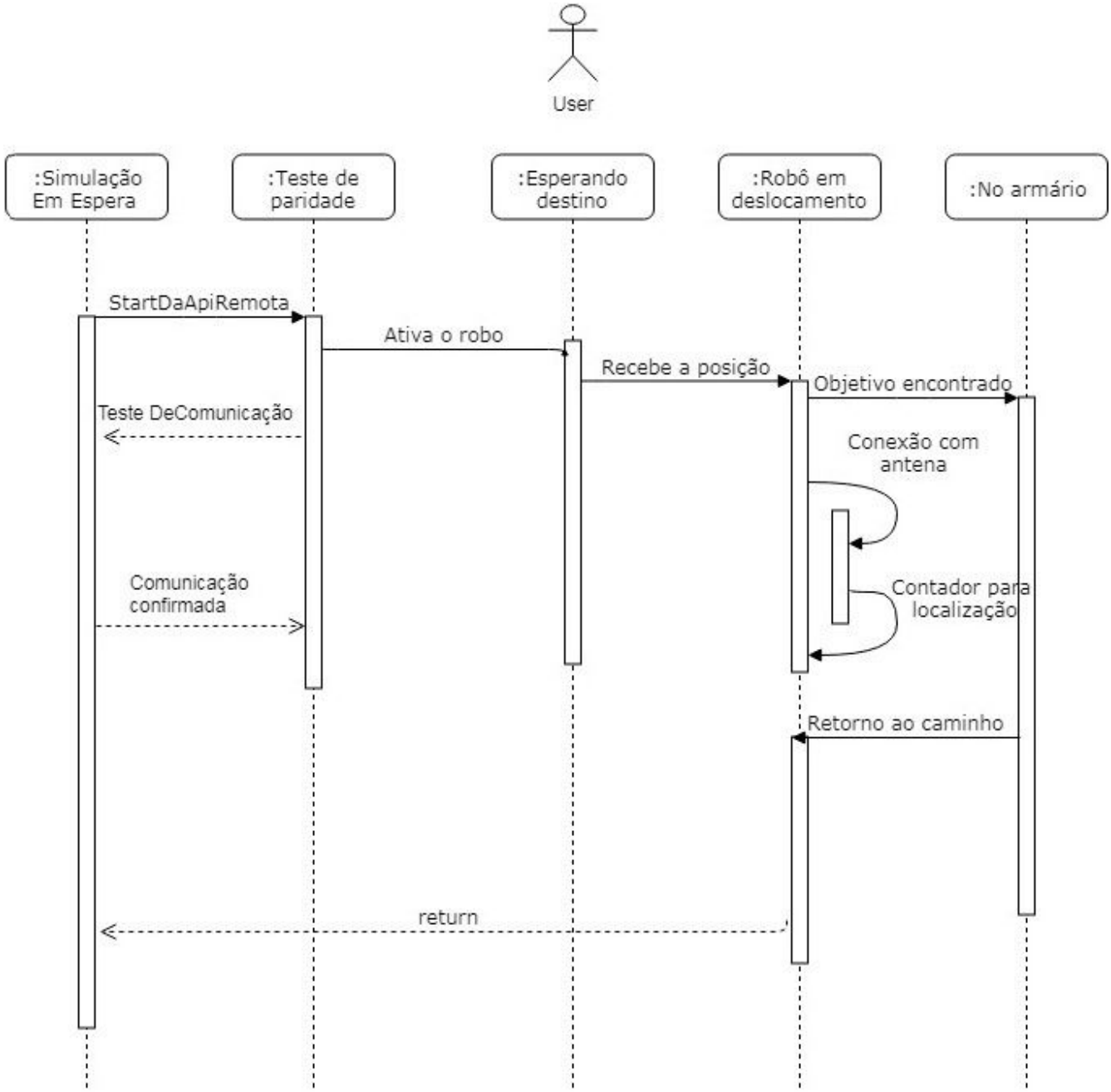
Figura 20: Lógica funcionamento do robô com comunicação NFC utilizando *dummy*



Fonte: Elaborado pelo próprio autor

Observa-se que seu comportamento final, que associa todos os programas descritos no capítulo, é representado pelo diagrama de sequência ilustrado na figura 21.

Figura 21: Diagrama de sequência utilizado na API e no script do V-REP



Fonte: Elaborado pelo próprio autor

4.6 Deslocamento e cinematica

Quando o robô realiza uma curva, as variáveis de inclinação, tempo de realização da curva e velocidade são fundamentais. Para se concluir a execução do programa o estudo da cinemática dos robôs é necessária, a qual envolve as variáveis citadas, sendo estas limitadas pelas dimensões e dinâmica dos robôs. Tais valores são vistos a seguir.

4.6.1 Propriedades do *BubbleRob*

O robô foi desenvolvido seguindo o tutorial do *BubbleRob*, suas propriedades, apresentadas na tabela 2, seguiram as recomendações do tutorial, e as proporções entre a massa, momento de inércia e torque buscou-se manter constante na execução dos próximos robôs.

Tabela 2. Valores das propriedades cinéticas do *BubbleRob* .

Parte do robô	Tamanho (x,y,z)[cm]	Massa [kg]	Momento de inercia [10^(-4)*m²]
Chassi	(20,20,20)	4.18	(4,4,4)
Rodas	(8,2,8)	0.8	(4.3,8,4.3)
Apoio	(5,5,5)	0.8	(4.3,8,4.3)
Distancia entre rodas		25cm	
Largura do ultrasônico		15cm	

4.6.2 Propriedades do TR1

Durante o as modificações no robô TR1, colocou-se uma maior massa nas rodas, mesmo essas ocupando um menor volume, isto se deu buscando dar mais estabilidade para o robô, sendo que chassi do robô ficou mais leve e o apoio continuou sendo uma esfera maciça, o´que facilitaria que o robô empina-se. Observa-se os valores definidos tabela 3.

Tabela 3. Valores das propriedades cinéticas do TR1.

Parte do robô	Tamanho (x,y,z)[cm]	Massa [kg]	Momento de inercia [10^(-4)*m²]
Chassi	(14.5,11.5,7.5)	2	(10,10,10)
Rodas	(5.5,0.46,5.5)	0.82	(11,11,11)
Apoio	(2.7,2.7,2.7)	1	(1.31,1.31,1.31)
Distancia entre rodas		13 cm	
Largura do ultrasônico		9 cm	

4.6.3 Propriedades do segundo robô

Observa-se que o robô não possui um terceiro apoio no chão dependendo da estrutura do chassi para se manter na vertical. Tal característica atrapalha sua locomoção. Observa-se as características de sua cinemática na tabela 4.

Tabela 4. Valores das propriedades cinéticas do segundo robô.

Parte do robô	Tamanho (x,y,z)[cm]	Massa [kg]	Momento de inercia [10^(-4)*m²]
Chassi	(15,10.5,21)	2.15	(1.7,1.6,1.2)
Rodas	(7.5,0.8,7.5)	0.182	(1.28,3.12,1.28)
Apoio	-	-	-
Distancia entre rodas		11.5 cm	
Largura do ultrasônico		7.5 cm	

4.6.4 Propriedades dos motores

O torque é a força que o motor exerce sobre o robô, que dependendo da massa e do momento de inércia do mesmo tem sua velocidade alterada até um valor máximo. As propriedades dinâmicas dos motores, vistos na tabela 5, são definidos no próprio V-REP, a velocidade alvo dos motores controlados pela API é igual a zero para que uma vez que a simulação é iniciada o robô esteja parado até receber algum valor de velocidade pelo comando da API.

Tabela 5. Valores das propriedades dos motores dos três robôs .

Robô	Velocidade alvo[deg/s]	Torque maximo [N]
BubbleRob	10	50
TR1	0	20
Segundo robô	0	10

4.6.5 Propriedades do armazém

O armazém, visto na figura 10, possui as dimensões da tabela a seguir, no qual observa-se que o corredor que oferece acesso a fileiras é mais largo que os demais.

Tabela 6. Descrição das distâncias que compõem o armazém

Armazem	Valor
Largura total	187 cm
Comprimento total	165 cm
Largura dos armarios	21 cm
Distância entre um armário e o da frente	23 cm
Largura dos corredores	23 cm
Largura do corredor pré-fileiras	30 cm

4.6.6 Valores para o programa

Observa-se na tabela 7 que o valor da inclinação, já citado neste capítulo, é consideravelmente baixo, isto se dá para realização de curvas mais fechadas no decorrer do armazém.

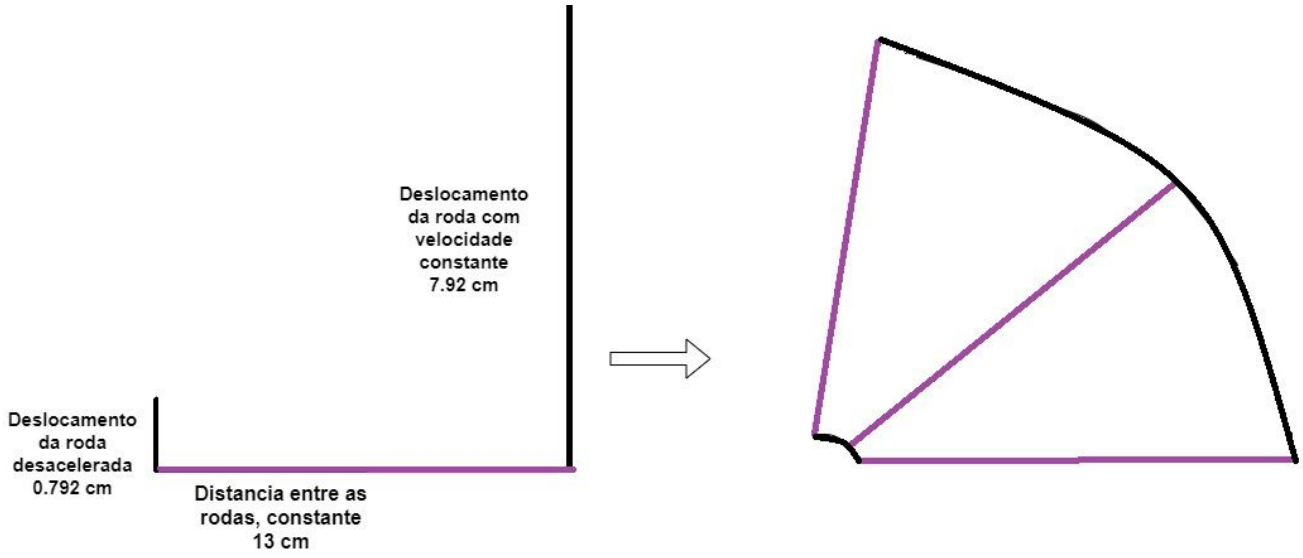
Tabela 7. Valores de controle de curva definidos na API.

Caracteristicas do programa	Valor
velocidade terminada	6 deg/s
Inclinação	0.1
tempo de curva	14 s

No estudo de caso do TR1, o robô demora quatorze segundos(t=14 s) para realizar uma curva de 90º. Isto se dá pelo fato que sua velocidade convertida em centímetro por segundo, isto é sua velocidade vezes o diâmetro da roda, é equivalente V=0.66cm/s. Isto resulta que em quatorze segundos um dos lados se desloca 7.92 cm, produto entre velocidade e tempo (D1 = V * t = 7.92 cm), enquanto o lado para qual deseja virar se desloca 0.792 cm, produto entre velocidade tempo e inclinação (D2 = V * t * 0.1 = 0.92 cm).

Tendo que a distância das rodas do TR1 é 13 cm, o robô realiza um movimento representado pela figura 22.

Figura 22: Representação da diferença do deslocamento entre as rodas e esta diferença ligada pela distância entre as rodas



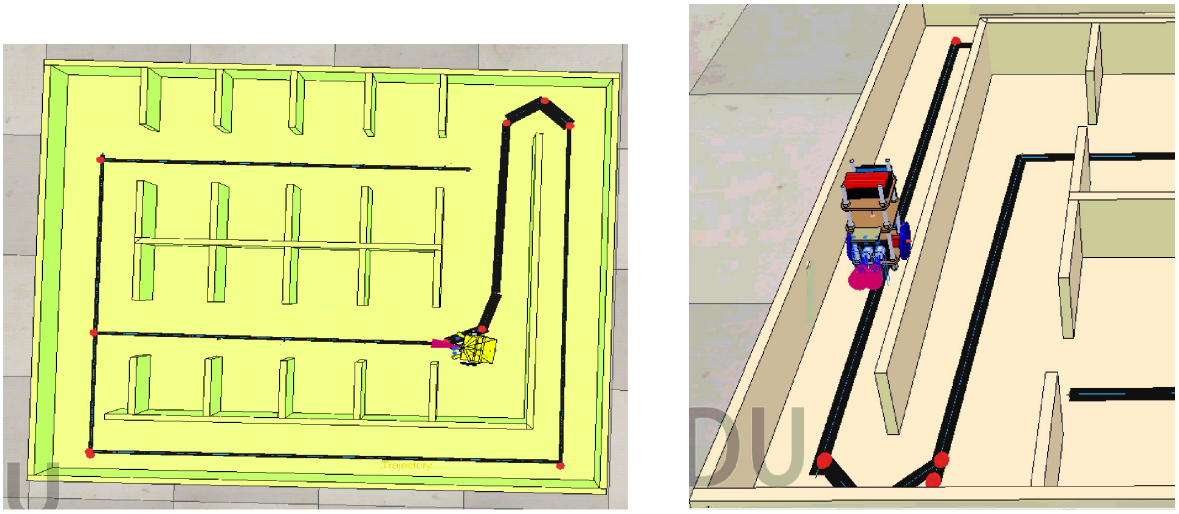
Fonte: Elaborado pelo próprio autor

5 RESULTADOS

O projeto se mostrou bem-sucedido com os um dos robôs testados, o TR1, enquanto no segundo a API teve sucesso de comunicação, porém seu deslocamento no armazém não ocorreu como esperado por inconsistências na adequação de variáveis cinemáticas, como massa e momento de inércia. Ambos possuem a mesma lógica na movimentação, com suas diferenças geométricas e dinâmicas sendo abrangidas pelo algoritmo.

O principal resultado desse trabalho consiste nas duas simulações representadas na figura 23, em que a API desenvolvida é testada nos dois robôs fornecidos para acesso à um dos compartimentos de armazenagem e retorno ao ponto de saída por meio de uma rota cíclica. Na imagem da direita a simulação aguarda a entrada de um valor para que o segundo robô inicie sua movimentação, enquanto na imagem da esquerda o TR1 já está em direção a uma célula de armazenamento.

Figura 23: Capturas de tela durante atividades de navegação em dois dos cenários implementados.

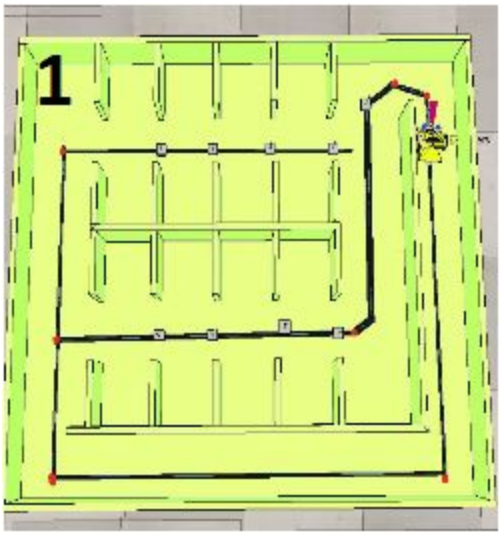
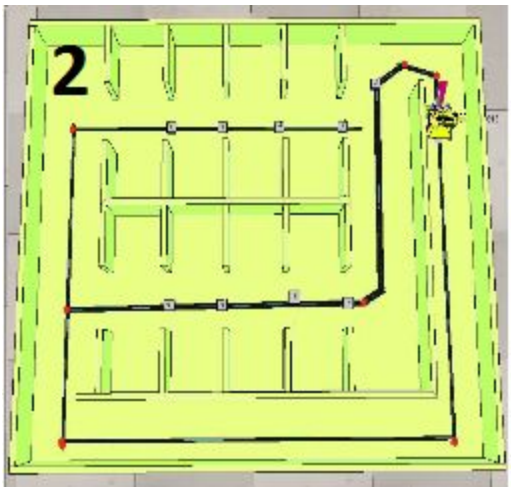


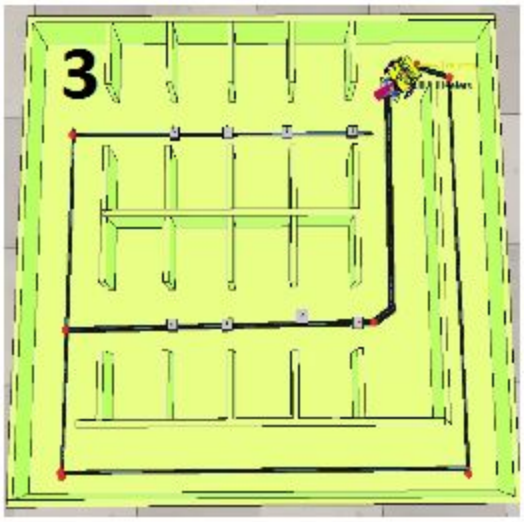
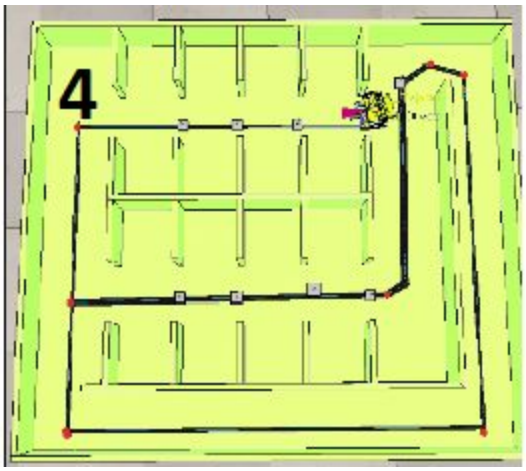
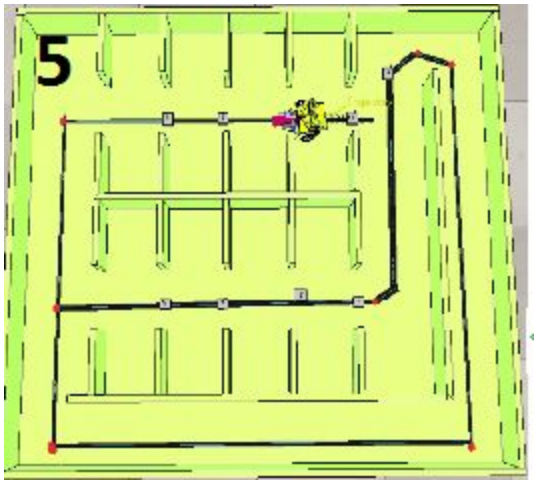
Fonte: Elaborado pelo próprio autor

5.1 Execução da simulação

Observa-se o resultado da execução do projeto através dos *prints* da simulação nas figuras 22 e 23, para as células de armazenagem 6 e 14, respectivamente. A célula 6 está posicionada no primeiro corredor, enquanto a célula 14 posiciona-se no segundo corredor. Uma descrição detalhada do movimento do robô em operação, ilustrado nas figuras 24 e 25.

Tabela 8. Fluxo de decisão para encontrar a célula de armazenamento 6.

Etapa	Descrição do movimento e valor das variáveis
	Inicio do programa. Contador(1) = 6
	Alteração das variáveis para realizar a primeira curva, onde é armazenado em uma variável auxiliar qual armazém é o destino. Aux= 6 Contador(1) = 1

	Realização da primeira curva e recebe da variável auxiliar o objetivo. Contador(1) = Aux = 6
	Recebe -2 ao passar pelo final da primeira curva. Um cuidado deve ser tomado para que o comando saia da ordem da primeira curva, antes da etiqueta NFC. Contador(1) = 4
	5 Recebe -2 ao passar pela etiqueta NFC. Contador(1) = 2

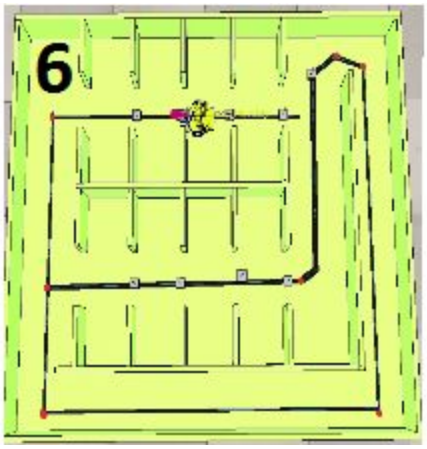
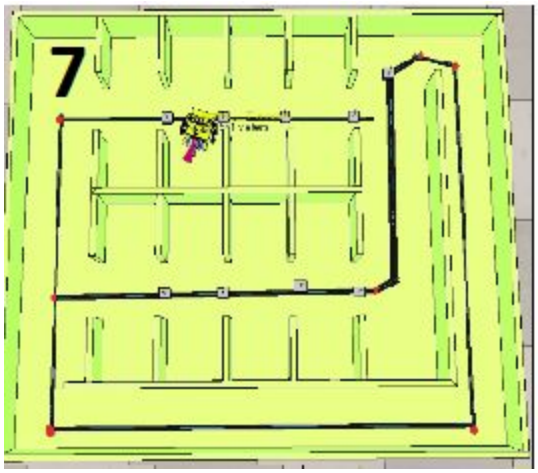
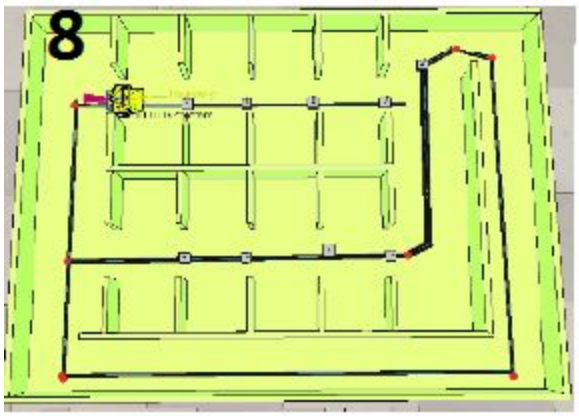
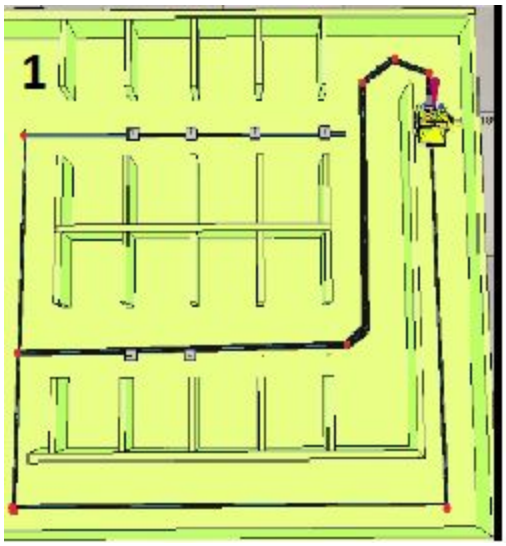
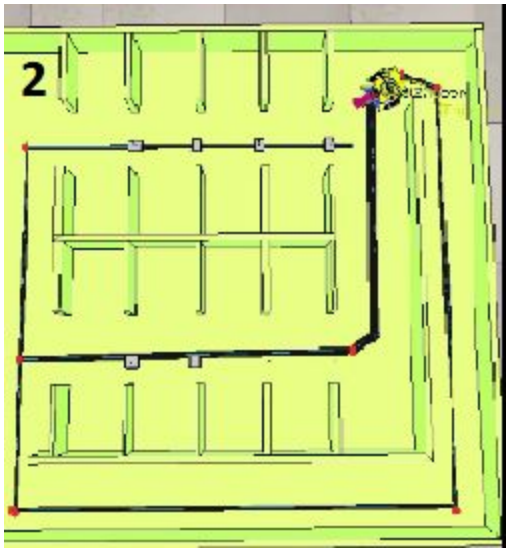
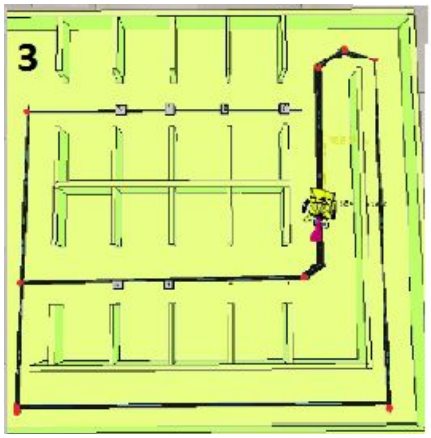
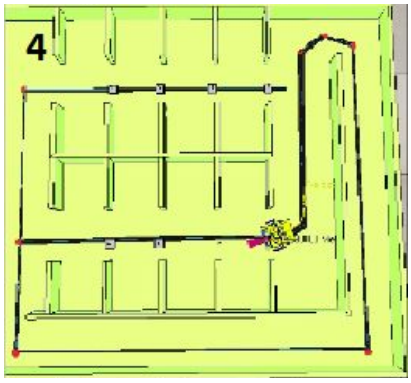
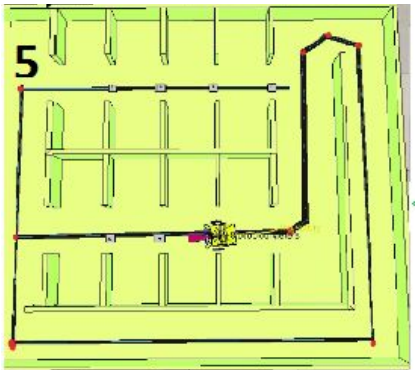
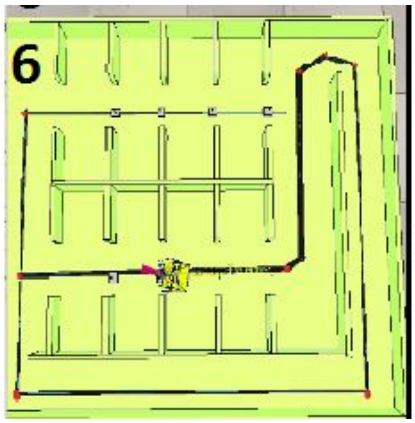
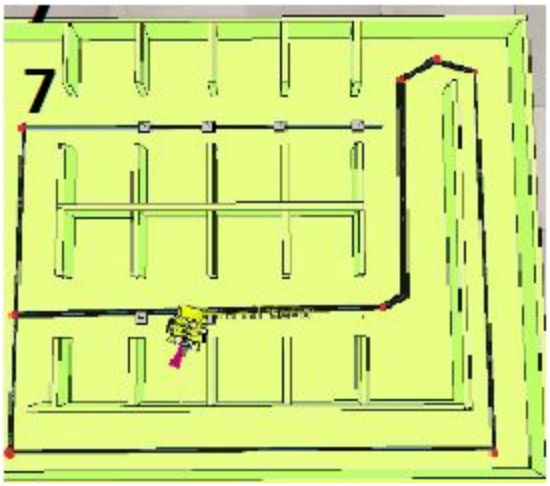
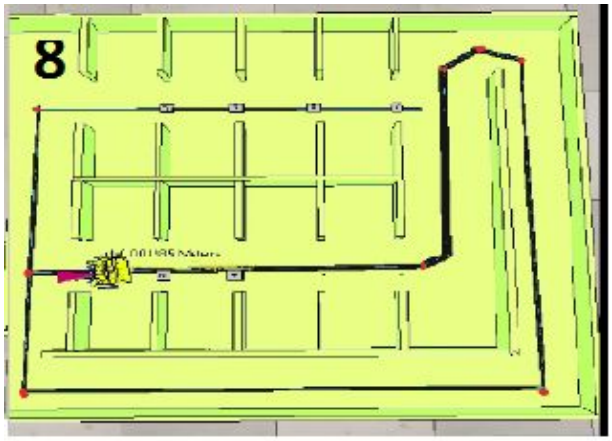
	Contage recebe -2 e se iguala a zero. Contador(1) = 0
	Curva realizada. Contador(1) = -2
	Retornar ao caminho. Contador(1) = -4

Tabela 9. Fluxo de decisão para encontrar o armário 14.

Etapa	Descrição do movimento e valor das variáveis
	Início do programa, Contador recebe -8 para compensar toda a fileira que foi ultrapassada e +2 para ignorar a etiqueta NFC Referente à primeira fileira. Contador(1) = 14 - 8 + 2 = 8
	Contador recebe -2. Contador(1) = 6

	Realização da curva na segunda fileira seguindo simplesmente a lógica do seguidor de linha. Contador(1) = 6
	Recebe -2 ao passar pela primeira etiqueta NFC da segunda fileira. Contador(1) = 4
	Recebe -2 ao passar pela segunda etiqueta NFC da segunda fileira. Contador(1) = 2
	Contagem recebe -2 ao passar pela terceira etiqueta NFC da segunda fileira e se iguala a zero. Contador(1) = 0

	Curva realizada. Contador(1) = -2
	Retornar ao caminho. Contador(1) = -4

A simulação em espera e o teste de paridade são realizados durante a imagem 1 da tabela 8 e 9. Esperar o destino para iniciar o deslocamento do robô ocorre na imagem 2. As imagens 3 a 6 correspondem ao robô em deslocamento e seu respectivo contato com as antenas de localização do armazém. A imagem 7 corresponde a identificação do armário requerido e após isto a imagem 8 indica o retorno do robô ao ponto de início.

6 CONCLUSÕES E PROPOSTAS DE TRABALHOS FUTUROS

Esta monografia apresentou a simulação de um modelo de armazém automatizado, tendo demonstrado no capítulo anterior os resultados encontrados. Tem-se a seguir uma análise sobre os objetivos estipulados no início do trabalho e recomendações para trabalhos futuros, estabelecendo possíveis contribuições.

6.1 Objetivos analisados

Analisando-se os objetivos citados na seção introdutória: Simular um ambiente de armazenamento autônomo e modular, divulgar funcionalidades do V-REP em ambiente acadêmico e buscar que esse ambiente seja funcional para dois robôs, observam-se os resultados alcançados.

O ambiente simulado oferece a base para a simulação de um armazém, considerando modelos de simulação de robôs móveis *open-source* já relatados na literatura, disponibilizando a movimentação em um armazém com acesso a 16 postos de armazenagem distintos. Este trabalho não aborda o mecanismo de carga ou descarga de um objeto no robô, focando-se apenas no processo de navegação com tomadas de decisão. A modularidade do ambiente é vista em sua estrutura sendo simples a indexação de outros armazéns, sendo necessário a atualização da comunicação NFC, como observado na figura 26.

A divulgação do software V-REP e suas funcionalidades se dará com a disponibilização dos códigos e dos resultados deste em ambiente online e gratuito. E para ambiente acadêmico, a divulgação se culminou na apresentação deste projeto no I Simpósio Goiano de Engenharia Mecanica (SIGMEC) (SANTOS; FONSECA, 2018).

Os testes mostraram que o ambiente e os robôs desenvolvidos são abrangentes, no que se refere ao problema proposto, também é facilmente extensível, como pode ser visualizado na figura 26, com o possível desenvolvimento de software customizado possivelmente utilizando a lógica da figura 20.

Ao concluir este trabalho, é perceptível que conceitos de engenharia de software, que se aplicados mais cedo, poderiam ter facilitado a organização do projeto e adiantado resultados obtidos. Pois, mesmo com a Coppelia Robotics oferecendo uma alta quantidade de tutoriais e fóruns, há uma vasta possibilidade de erros e prováveis *bugs* da simulação que podem bloquear totalmente o desenvolver do projeto.

No demais conhecimentos relacionados à engenharia de software cumprem seu propósito quando bem aplicados, orientando programadores a estabelecerem melhores metodologias para desenvolvimento. Conhecimentos relacionado a mecanismos de programação orientada objeto fazem parte estrutura da simulação sendo uma aplicação prática de computação gráfica, na qual o *V-REP* automatiza tal parte para o usuário.

Analisando os objetivos encontrados, percebem-se limitações relacionadas ao seu desenvolvimento, tais limitações tornam o ambiente simulado algo ainda distante de uma aplicação real.

6.2 Trabalhos futuros

Dentre as sugestões de trabalho futuro, destacam-se a expansão do número de robôs, tornar mais realista o cenário, melhoria da precisão da locomoção e melhorar a geometria do robô possibilitando atingir-se maior velocidade no deslocamento, tendo em vista que para que o robô alcance um armário na primeira fileira, como da figura 24, são necessários por volta de dois minutos e meio do tempo de simulação. Alguns projetos que poderiam expandir o desenvolvido seriam:

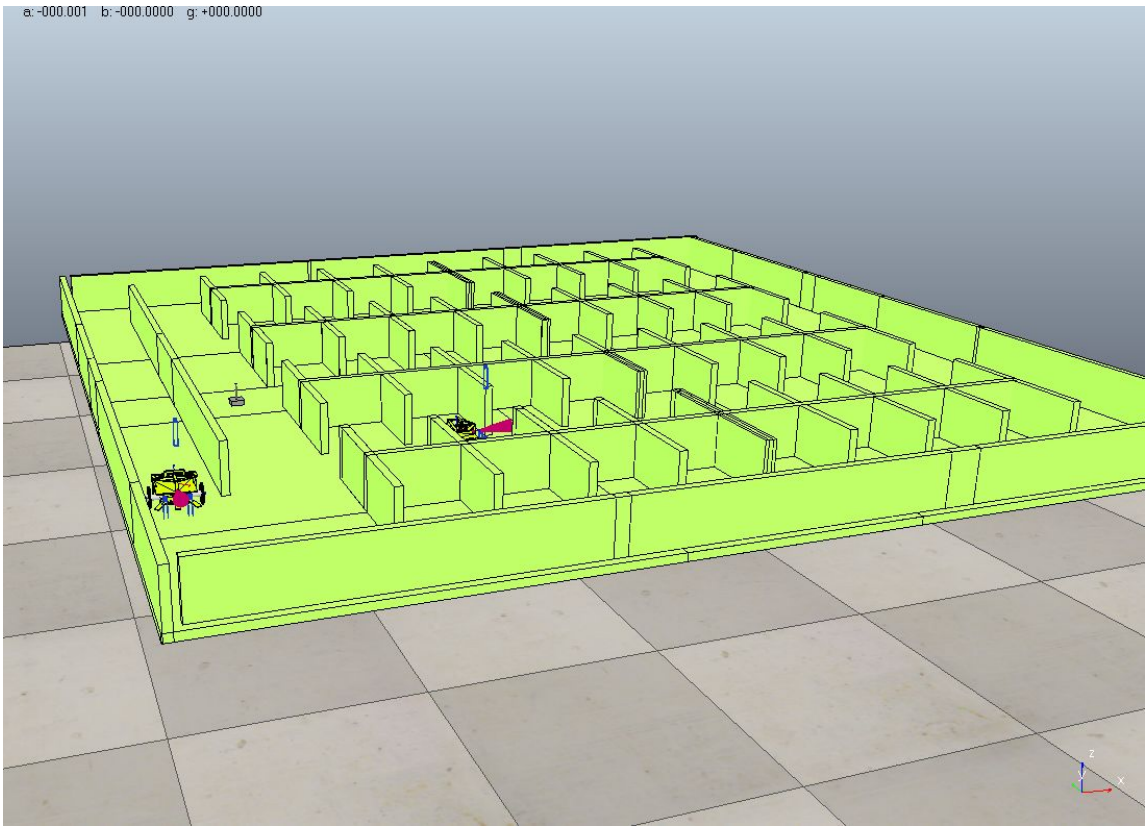
Como trabalhado por Ying Tan (2017) o trabalho com multiagentes é uma realidade no desenvolvimento de qualquer ambiente real, o mesmo trabalhou com medidas de segurança viáveis em sua pesquisa, porém, pela estrutura desenvolvida seria mais aplicável para segurança um controle através de inteligência externa que possa interromper a função de um ou mais robôs.

Outra otimização viável seria a inserção de funcionários no armazém, fazendo assim um teste relacionado à segurança e adequação do sistema para um ambiente de robótica colaborativa, colaborando no trabalho citado na fundamentação teórica de Inam (2018) sobre

segurança para robôs e humanos.

Seria de suma importância futuramente o teste em espaço físico, como ambos robôs testados são inspirados em robôs reais, assim como descrito na seção 2. Tal teste poderia contribuir com a pesquisa de COLLINS (2019) que analisou o acúmulo de erro entre simulação e realidade, assim como no sentido de solidificar e embasar o funcionamento do V-REP.

Figura 24: Modelo de ambiente com 64 armarios



Fonte: Elaborado pelo proprio autor

REFERÊNCIAS

ADVFN. **Grafico da AMAZON**. Disponível em: https://br.advfn.com/bolsa-de-valores/nasdaq/AMZN/grafico. Acesso em: 15 nov. 2019.

ARTIMA. **The Making of Python A Conversation with Guido van Rossum, Part I by Bill Venners**. 2003, Disponível em: https://www.artima.com/intv/python.html. Acesso em: 6 jun. 2019.

BARROS, João O. , **V-REP API framework – ROS interface**, 5 de março de 2014, Disponivel em: encurtador.com.br/druAD. Acessado em: 06 dez 2019.

BOEHM, B. **Um modelo espiral de desenvolvimento e aprimoramento de software,** 1988, IEEE Computer, vol.21, , pp 61-72

Carros autônomos **,** Nerdologia Tech, Direção: Átila Iamarino, Produção: Paulo Silveira. Estúdio 42, Video(10 min), port, Disponivel em: https://www.youtube.com/watch?v=kWf4ZFO78qE. Acessado em: 06 nov 2019.

COLLINS, J.; HOWARD, D.; LEITNER, J. **Quantificando a diferença de realidade em tarefas de manipulação robótica**. *In*: *International Conference on Robotics and Automation*, 2019, Montreal.

COPPELIA ROBOTICS, **BØ-based remote API**, 2019b, online, Disponivel em :http://www.coppeliarobotics.com/helpFiles/en/b0RemoteApiOverview.htm. Acessado em: 09 nov 2019.

COPPELIA ROBOTICS, **Crie. Componha. Simule. Qualquer Robô**, 2019a, online, Disponivel em : http://www.coppeliarobotics.com. Acessado em: 09 nov 2019.

COPPELIA ROBOTICS, **BubbleRob tutorial**, 2019c, online, Disponivel em : http://www.coppeliarobotics.com/helpFiles/en/bubbleRobTutorial.htm. Acessado em: 12 fev 2019.

COPPELIA ROBOTICS, **Dummies**, 2019d, online, Disponivel em : http://www.coppeliarobotics.com/helpFiles/en/dummies.htm. Acessado em: 09 nov 2019.

COPPELIA ROBOTICS, **Remote API functions (Python)**, 2019e, online, Disponivel em : http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm. Acessado em: 09 nov 2019.

COPPELIA ROBOTICS, **Escrevendo código dentro e fora do V-REP**, 2019f, online, Disponivel em : http://www.coppeliarobotics.com/helpFiles/en/writingCode.htm. Acessado em: 14 nov 2019.

COPPELIA ROBOTICS, **Legacy remote API, S. D.,** online**,** disponivel em: http://www.coppeliarobotics.com/helpFiles/en/legacyRemoteApiOverview.html. Acessado em: 25 jul 2019.

COPPELIA ROBOTICS. **Virtual Robot Experimentation Platform User Manual**, online, Disponivel em :encurtador.com.br/fzEGV Acessado em: 06 dez 2019.

COSTA, P.A., **SIMULAÇÃO DO COMPORTAMENTO DE UM ROBÔ MÓVEL NÃO COMERCIAL EM AMBIENTE DE ARMAZÉM 2D**, Simpósio Goiano de Engenharia Mecânica (SIGMEC) (Santos; Fonseca, 2018), 2019.

DEARO Guilherme**, Amazon é a marca mais valiosa do mundo, revela Brand Finance 2019,** EXAME, ONLINE, 23 jan 2019, 16h06 , Disponivel em: https://bit.ly/2WRvlbc. Acessado em: 06 dez 2019.

FERREIRA, M. V. M. FONSECA, J. P. S.; TAVARES, J. J. P. Z. S. "**Attabot: Open Platform Inspired on Brazilian Ants for Swarm Robots**, *in*: 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE),pp. 225-229.doi: 10.1109/LARS/SBR/WRE.2018.00049, João Pessoa, 2018.

FOLDOC, **Application Program Interface,** 1995, Disponivel em: http://foldoc.org/Application+Program+Interface. Acessado em : 09 dez 2019

FONSECA, J. P. S.**Redes de Petri de alto nível e PNRD invertida associadas ao controle de robôs móveis: uma abordagem para operações de busca e salvamento em trilhas e travessias**. 146 f. Tese de Doutorado – Universidade Federal de Uberlândia, Minas Gerais, Brazil, 2018.

FONSECA, J. P. S.; SILVA, C. E. A.; SOUSA, A. R.; TAVARES, J. J. P. Z. S. **Strategies for search and rescue multirobot system based on place/transition Petri nets and RFID distributed database.** In: 2017 LATIN AMERICAN ROBOTICS SYMPOSIUM (LARS) AND 2017 BRAZILIAN SYMPOSIUM ON ROBOTICS (SBR), 2017, Curitiba. Proceedings... p. 1- 6.

GASPAROTTO, Henrique Machado , **Os 4 pilares da Programação Orientada a Objetos**, ONLINE, 2014 , Disponível em: encurtador.com.br/cDFS0. Acessado em: 06 dez 2019.

BIONDO, Giovani. **UM PROCESSO DE CONVERSÃO DE SISTEMAS LEGADOS PROCEDURAIS PARA ORIENTADO A OBJETOS, DIRECIONADO PELA ARQUITETURA MVC.** UNIVERSIDADE DE CAXIAS DO SUL**,** 2017. BENTO GONÇALVES, v. 1, n. 1, p. 16-16, Disponível em:https://repositorio.ucs.br/xmlui/handle/11338/3809. Acesso em: 09 dez. 2019.

BLOGLOGÍSTICA, **A TECNOLOGIA DOS CENTROS DE DISTRIBUIÇÃO DA AMAZON,** 2015, Disponivel em: encurtador.com.br/ALTX0. Acesso em: 09 dez 2019.

INAM, Rafia, **Segurança para o Armazém Automatizado exibindo robôs colaborativos,** 28a Conferência Europeia de Segurança e Confiabilidade (ESREL'18), 2018.

JJROMEROMARRAS, **V-REP utilización del API Remota, 2016,** online, disponivel em: encurtador.com.br/dsBG3. Acessado em: 01 out 2019.

JUNGTHON Gustavo; GOULART, Cristian Machado ,**Paradigmas de Programação,** 2010, 8p, Faculdade de Informática de Taquara (FIT), Disponível em: encurtador.com.br/aBEV1. Acessado em: 06 dez 2019.

KUMMER, Nikolai, **VREP 04 – Connecting VREP To Python,** 13 de jan. de 2015, Disponivel em: http://34.208.13.223/VREP/04PythonTutorial. Acessado em: 01 jun 2019.

LOBO, Alexandre, **Fórum de Davos e os robôs na logística**, ILOS - Especialistas em Logística e Supply Chain, 2016, ONLINE, Disponível: https://www.ilos.com.br/web/forum-de-davos-e-os-robos-na-logistica. Acessado em: 06 nov 2019.

LUA.ORG, **A Linguagem de Programação Lua,** 2018, PUC-Rio:LUA, Disponível em: https://www.lua.org/portugues.html. Acessado em: 06 jul 2019.

MARTINS João Pavão , **Programação em Python - Introdução à Programação Utilizando Múltiplos Paradigmas**, ed.1, Universidade T´ecnica de Lisboa, 2012.

POLLUX, **Case Amazon: Uso de robôs na logística interna fez dela a 3ª empresa mais valiosa do mundo!,** 2018, Beatriz disponivel em: encurtador.com.br/nMRT1. Acessado em: 06 nov 2019.

ROGGIA, Leandro; FUENTES, Rodrigo Cardozo, **Automação Industrial,** e-tec Brasil, pp. 14-21**,**Disponivel em: encurtador.com.br/dgy26. Acessado em: 06 nov 2019.

RONSZCKA f. a., **Método para a criação de linguagens de programação e compiladores para o paradigma orientado a notificações em plataformas distintas,** 2019.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots. Cambridge**: The MIT Press, 2004. 335 p. T2

SILVA, A. G. **Projeto e construção de robô móvel tipo micromouse para bancada de busca e salvamento**. . 63 f. Monografia de Conclusão de Graduação, Universidade de Federal de Uberlândia, Uberlândia. 2017.

STACKOVERFLOW, **Resultados da pesquisa do desenvolvedor 2018,** 2018, Online, Disponível em: encurtador.com.br/glJL6. Acessado em: 06 dez 2019.

TAN, Ying: ZHENG, Zhonhg-yang, **Avanço da pesquisa em robótica de enxame,** 2013, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

TECMUNDO, **Amazon já tem mais de 100 mil robôs autônomos em seus galpões,** 2018, Online, Disponivel em: encurtador.com.br/ehtCS. Acessado em: 06 dez 2019.

Weisbin, C.R. and D. Lavery, **NASA rover and telerobotics technology**, IEEE Robotics & Automation Magazine, pp. 14-21 vol. 1, no. 4, 1994.

WINICK, Erin. **O segredo para parar o apocalipse do robô? Manteiga de pipoca.**, 2018, Disponivel em: encurtador.com.br/qyzMN. Acessado em: 09 dez 2019.

APÊNDICE 1

Codigos do simulador

Codigo associado ao Dummy

```
function sysCall_init()
    i=0
end

function sysCall_cleanup()
end

function sysCall_sensing()
    if (sim.boolAnd32(i,1)==1) then
        messageToSend="Marco"..i

sim.sendData(sim.handle_all,0,"marco-polo",messageToSend,sim.getObjectHandle("Antenna
"),2,3.1415*20/180,3.1415*40/180)
        --sim.addStatusbarMessage(sim.getScriptName(sim.handle_self).." just sent
'"..messageToSend.."')
    end
    i=i+1

    data=sim.receiveData(0,"marco-polo",sim.getObjectHandle("Antenna"))
    while (data) do
        sim.addStatusbarMessage(sim.getScriptName(sim.handle_self).." just received
'"..data.."')
        data=sim.receiveData(0,"marco-polo",sim.getObjectHandle("Antenna"))
    end
end
```

APÊNDICE 2

Codigos final em Lua antes da API

```
function sysCall_init()

    TR1Base=sim.getObjectAssociatedWithScript(sim.handle_self)
    leftMotor=sim.getObjectHandle("Revolute_esq")
    rightMotor=sim.getObjectHandle("Revolute_dir")
    noseSensor=sim.getObjectHandle("TR1_sensingNose")
    minMaxSpeed={50*math.pi/180,300*math.pi/180}
    backUntilTime=-1 -- Tells whether bubbleRob is in forward or backward mode
    floorSensorHandles={-1,-1,-1}
    floorSensorHandles[1]=sim.getObjectHandle("Sensor_esq")
    floorSensorHandles[2]=sim.getObjectHandle("Sensor_mid")
    floorSensorHandles[3]=sim.getObjectHandle("Sensor_dir")
    -- Create the custom UI:
    xml = '<ui title="'..sim.getObjectName(TR1Base)..' speed" closeable="true"
resizeable="false" activate="false">'..[[
                <hslider minimum="0" maximum="100" on-change="speedChange_callback"
id="1"/>
            <label text="" style="* {margin-left: 300px;}"/>
        </ui>
        ]]
    ui=simUI.create(xml)
    speed=(minMaxSpeed[1]+minMaxSpeed[2])*0.5

simUI.setSliderValue(ui,1,100*(speed-minMaxSpeed[1])/(minMaxSpeed[2]-minMaxSpeed[1]))

end
```

```
function speedChange_callback(ui,id,newVal)
    speed=minMaxSpeed[1]+(minMaxSpeed[2]-minMaxSpeed[1])*newVal/100
end

function sysCall_actuation()
    result=sim.readProximitySensor(noseSensor)
    if (result>0) then backUntilTime=sim.getSimulationTime()+8 end
    -- read the line detection sensors:
    sensorReading={false,false,false}
    for i=1,3,1 do
        result,data=sim.readVisionSensor(floorSensorHandles[i])
        if (result>=0) then
            sensorReading[i]=(data[11]<0.3) -- data[11] is the average of intensity of the image
        end
    end
    -- compute left and right velocities to follow the detected line:
    rightV=speed
    leftV=speed

    if sensorReading[3] then
        rightV=0.001*speed
    else
    if sensorReading[1] then
        leftV=0.01*speed
    end
    end
    --if sensorReading[1] and sensorReading[3] then
    --   backUntilTime=sim.getSimulationTime()+2
    --end
    if (backUntilTime<sim.getSimulationTime()) then
```

```
        sim.setJointTargetVelocity(leftMotor,leftV)
        sim.setJointTargetVelocity(rightMotor,rightV)
    else
        if(backUntilTime>(sim.getSimulationTime()+4)) then
            if sensorReading[3]then
            backUntilTime=-1
            end
            if sensorReading[1]then
            backUntilTime=-1
            end
        sim.setJointTargetVelocity(leftMotor,speed/6)
        sim.setJointTargetVelocity(rightMotor,speed/3)
            else
            if sensorReading[3]then
            backUntilTime=-1
            end
            if sensorReading[1]then
            backUntilTime=-1
            end
            sim.setJointTargetVelocity(leftMotor,speed/1.4)
            sim.setJointTargetVelocity(rightMotor,speed/3)
        end
    end
end

function sysCall_cleanup()
    simUI.destroy(ui)
end
```

Código da API

Código do endereçamento da cena

```
function sysCall_threadmain()
    -- Put some initialization code here
    -- Put your main loop here, e.g.:
    -- while sim.getSimulationState()~=sim.simulation_advancing_abouttostop do
    --     local p=sim.getObjectPosition(objHandle,-1)
    --     p[1]=p[1]+0.001
    --     sim.setObjectPosition(objHandle,-1,p)
    --     sim.switchThread() -- resume in next simulation step
    -- end
end

function sysCall_cleanup()
    -- Put some clean-up code here
end
simRemoteApi.start(19999)
```

codigo em Python

```
Pcurva=0

if(Armazem<8):
   Pcurva=Armazem
   Armazem=12
   AntenaNFC= ’  ’

def Testesensor(Leitura):
    global Armazem
```

```
    cor=list(Leitura[2])
    if(len(cor)==1):
      cor=list(cor[0])
    leu=False

    # Armazem=7  #de 16posições
    if(len(Leitura)==5):
        leu=bool(Leitura[1])
    else:
            if(AntenaNFC[0] == 'D'):
             print(AntenaNFC)
             contador = int(AntenaNFC[1:])
             if(contador!=Armazem):
                 print('desiqual',Armazem)
                 Armazem=contador
        if(len(Leitura)==3):
             #print(cor)
             #print('Valores da cor')

tonalidade_branco=cor[0]*cor[1]*cor[2]*cor[3]*cor[4]*cor[5]*cor[6]*cor[7]*cor[8]*cor[9]*cor[10]*
cor[11]*cor[12]*cor[13]*cor[14]

tonalidade_preto=cor[0]+cor[1]+cor[2]+cor[3]+cor[4]+cor[5]+cor[6]+cor[7]+cor[8]+cor[9]+cor[1
0]+cor[11]+cor[12]+cor[13]+cor[14]
             if((tonalidade_branco==1.000)):
                 Armazem=Armazem-2
                 print('leu branco',Armazem)
             if((tonalidade_preto<4.000)):
                 leu=True
        else:
```

```
            print('erro no sensor') #apenas para testar o codigo

    return leu


def MudarVel(vele,veld):
    errrorCode=vrep.simxSetJointTargetVelocity(clientID,esq_motor_handle, vele,
vrep.simx_opmode_streaming)
    errrorCode=vrep.simxSetJointTargetVelocity(clientID,dir_motor_handle, veld,
vrep.simx_opmode_streaming)



if clientID!=-1:
        print ('Conectado ao servidor de API remota, funfo')
        vel=5
        tArm=5/vel
        inc=0.2

vrep.simxGetIntegerParameter(clientID,vrep.sim_intparam_mouse_x,vrep.simx_opmode_stre
aming) # Initialize streaming
        errrorCode=vrep.simxSetJointTargetVelocity(clientID,esq_motor_handle, 0,
vrep.simx_opmode_streaming)
        errrorCode=vrep.simxSetJointTargetVelocity(clientID,dir_motor_handle, 0,
vrep.simx_opmode_streaming)
        startTime=time.time()

        while(True):
            while ((time.time()-startTime) < 20):
                Leitura_esq=vrep.simxReadVisionSensor(clientID,
sensor_esq,vrep.simx_opmode_oneshot_wait)
```

```
            Leitura_dir=vrep.simxReadVisionSensor(clientID,
sensor_dir,vrep.simx_opmode_oneshot_wait)
            Leitura_mid=vrep.simxReadVisionSensor(clientID,
sensor_mid,vrep.simx_opmode_oneshot_wait)
            Leitura_nose = vrep.simxReadProximitySensor (clientID,
sensor_handle,vrep.simx_opmode_oneshot_wait)
            print('tempo no armazem(seguir linha)',(time.time()-startTime))
            if(Armazem==0|Armazem==1):
                startTime=startTime-50



            if(Testesensor(Leitura_nose)):
              MudarVel(-(vel*inc),-(vel*inc*inc))
            else:
              if (Testesensor(Leitura_esq)):
                  MudarVel(vel*inc,vel)
                  print('leu esquerdo')
              else:
                  if (Testesensor(Leitura_dir)):
                     MudarVel(vel,vel*inc)
                     print('leu direito')
                  else:
                     MudarVel(vel,vel)

            if((time.time()-startTime)<0):
                startTime=startTime-1.36

            startTime=startTime+1.36
            time.sleep(0.1)
        vrep.simxAddStatusbarMessage(clientID,'Hello V-REP!',vrep.simx_opmode_oneshot)
```

```
while (time.time()-startTime>20):
    print('tempo no armazem(leu branco)',(time.time()-startTime))

    if(Armazem==0):
        print('armazem=0')
        if((time.time()-startTime)<(50+(tArm*0.5))):
            MudarVel(vel*inc,vel)
        else:
            if((time.time()-startTime)<(50+(tArm-1))):
                MudarVel(-vel*inc,-vel)
    if(Pcurva>0): #realizando a curva
        if(Armazem==1 |Armazem==-1 ):
            print('armazem=1 curva')
            Armazem=Pcurva

            if((time.time()-startTime)<(50+(tArm*0.5))):
                print('muda vel curva')
                MudarVel(vel,vel*inc)
            else:
                print('muda tempo curva')
                startTime=startTime+(50+tArm)

    else:
         if(Armazem==1 |Armazem==-1 ):
            print('armazem=1')
            if((time.time()-startTime)<(50+(tArm*0.5))):
                MudarVel(vel,vel*inc)
            else:
                if(Pcurva>0):
```

```
                    print('primeira curva')

                    print(Armazem)
                    startTime=startTime+(tArm*0.5)
                    print('mudo tempo')
                else:
                    if((time.time()-startTime)<(50+(tArm-2))):
                        MudarVel(-vel,-vel*inc)
            time.sleep(1)

            if(((time.time()-startTime)>(50+tArm))):
                Pcurva=0
                print('saindo do while')
                MudarVel(0,0)
                print('então saiu')
                startTime=time.time()


else:
   print('conexão falhou')
sys.exit('nâo deu')
```

ANEXO 1

Tipos de Chamada de função

Chamada de função de bloqueio

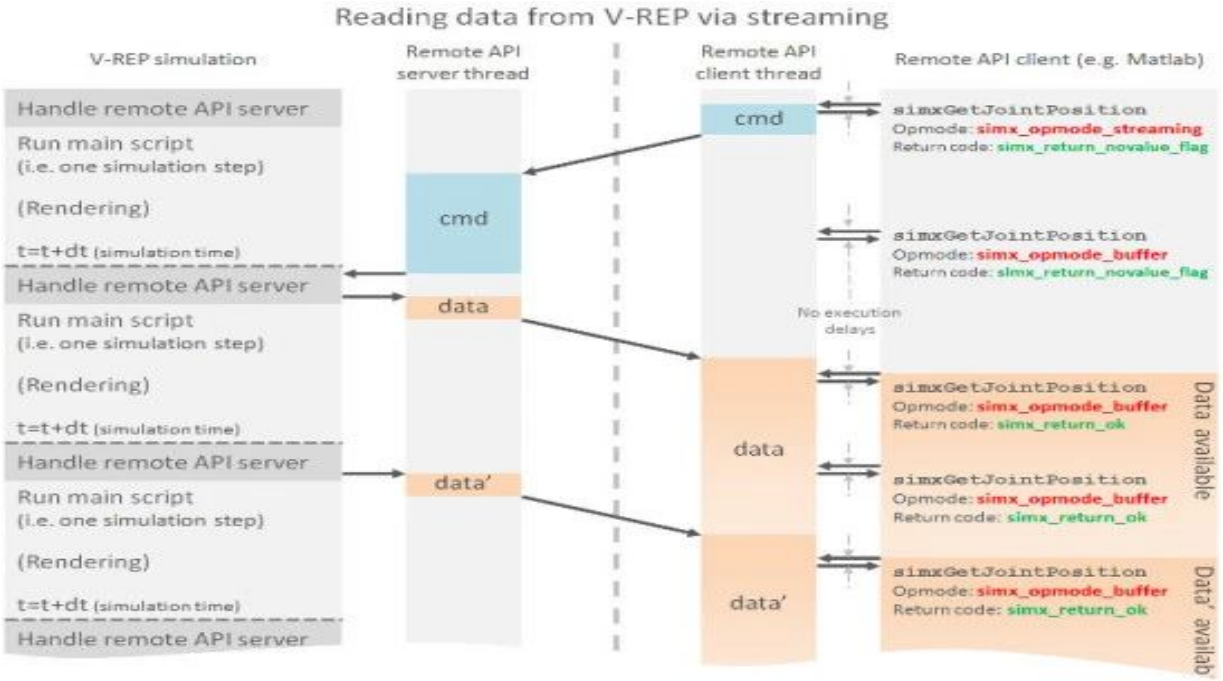
Neste tipo de execução é usado quando devemos aguardar a resposta do comando do servidor. Ou seja, quando um comando é executado indicando que seu modo está bloqueando, ele aguardará a resposta do servidor chegar, para que nosso código seja parado nessa chamada.

Chamada de funtion sem bloqueio

Ao contrário do anterior, as chamadas não estão bloqueando. Usando nas situações em que simplesmente queremos enviar informações para o servidor sem importar sua resposta.

Transmissão de dados

Um dos problemas ao usar a API remota é o atraso nas comunicações(esse *delay* é gerenciado pela comunicação baseada em um comando de *time.sleep,* mais detalhes fornecidos pela Copperlia Robots). Dependendo desse atraso e de nossa implementação, os resultados podem não estar corretos. Por exemplo, suponha que, com base nas informações dos sensores do robô,



Operação Síncrona

As chamadas para as funções da API remota serão executadas no modo assíncrono por padrão. Isso significa que a simulação do V-REP será executada independentemente do nosso programa cliente. Ou seja, não importa quais cálculos ou processos estamos fazendo em nosso programa que a simulação "seguirá seu curso".